

COMPUTABILITÀ

Vedremo

- procedura effettiva e funzione calcolabile
 - caratteristiche di un algoritmo / modulo di calcolo
 - questo ci basta per dedurre
 - esistenza di funzioni non calcolabili
- } informalmente

* Algoritmo o procedura effettiva

Introduzione informale! Vedremo poi come può essere formalizzata per mezzo di una sorta di computer idealizzato

Di algoritmi (anche se non sempre usiamo questo termine per riferirli) ne incontriamo in continuazione...

Fini dalle elementari, non ci viene insegnato solo che dati due numeri esiste la loro somma.

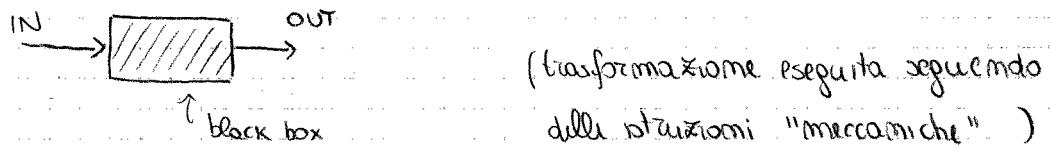
ma anche un procedimento per calcolare tale somma!

Algoritmo: descrizione di una sequenza di passi da seguire (in modo meccanico, senza intelligenza!) per realizzare un certo obiettivo
 trasformare input \rightsquigarrow output (con proprietà desiderate)
 (es: ingredienti \rightsquigarrow torta)

Esempi

- 1) dato $m \in \mathbb{N}$ dire se è primo
- 2) trovare il m mo numero primo
- 3) dividere un polimorfismo
- 4) $\sqrt{}$
- 5) mcm, MCD

Dunque possiamo pensare ad un algoritmo come ad una black box



se l'algoritmo è deterministico individua una funzione da valori di input a valori di output

Si dice che

- l'algoritmo calcola questa funzione e che
- la funzione è calcolabile
(effettivamente) |
 computabile

Dunque

* una funzione è computabile se esiste un algoritmo che la calcola
(ma potremmo non sapere qual è !!!)

Secondo questa nozione informale ciualmente sono calcolabili

- $\text{MCD}(x,y)$ massimo comum divisore (Euclide!)
- $f(m) = \begin{cases} 1 & m \text{ primo} \\ 0 & m \text{ non è primo} \end{cases}$
- $f(m) = m^{\text{ma}} \text{ numero primo}$

e anche

$$f(m) = m^{\text{ma}} \text{ cifra nell'espansione decimale di } \pi$$

infatti, da analisi sappiamo che

- esistono serie che convergono a π
- « tecniche per ottimizzare (per eccesso) l'errore causato
- del truncamento di una serie
- dall'aristotomamento nel calcolo del valore della serie
Troncato.

* Che dice invece della funzione

$$g(m) = \begin{cases} 1 & \text{se ci sono esattamente } m \text{ "5" consecutivi} \\ & \text{nell'espansione decimale di } \pi \\ 0 & \text{altrimenti} \end{cases}$$

Se ad un certo punto nello sviluppo di π troviamo m "5" consecutivi possiamo concludere $g(m) = 1$

ma a nessuno stadio, non avendo trovato gli m "5", possiamo escludere che compaiano nel seguito!

Qualcosa del tipo: calcola le cifre nell'esp. di π

- se trovo m "5" consecutivi $\rightsquigarrow g(m) = 1$
- altrimenti $\rightsquigarrow g(m) = 0$

NON è una procedura effettiva!

Note: Questo non significa che $g(m)$ è calcolabile ovvero che una procedura effettiva non possa essere trovata
(ad es sulla base di proprietà di π)

Al momento questa procedura non è nota! (almeno a me!)

* Un esempio molto più sottile

$$g(m) = \begin{cases} 1 & \text{se esiste una sequenza di almeno } m \text{ "5" consecutivi} \\ 0 & \text{altrimenti} \end{cases}$$

Calcolabile? Sì!

Infatti detto

$$k = \sup \{ m \mid \text{esisto } m \text{ "5" consecutivi in } \pi \}$$

c'sono due possibilità

$$\begin{array}{l} \text{e } k \text{ finito} \\ \Rightarrow h(m) = \begin{cases} 1 & m \leq k \\ 0 & \text{altrimenti} \end{cases} \end{array}$$

$$\stackrel{?}{=} k = \infty \quad \Rightarrow \quad h(m) = 1 \quad \forall m$$

In entrambi i casi la funzione h è ovviamente calcolabile!

1 - dato m , guarda se $m \in K$

si $\rightarrow 1$

No $\rightarrow 0$

2 - ignora m e restituisci 1

In questo paragrafo emergerà un problema: non conosciamo K !!

degli infiniti algoritmi menzionati non sappiamo quale sia quello giusto... ma sappiamo che ce n'è uno!

Ri - Nota: Affinché una funzione sia calcolabile occorre che esista un algoritmo che la calcola (ma non che noi sappiamo quale è!)

* Caratteristiche di un algoritmo (e del modulo di calcolo)

... affinché catturi l'idea intuitiva di procedura effettiva
 (che per un informatico significa che deve essere realizzabile, implem.
 in una macchina)

⇒ Un algoritmo è una sequenza di istruzioni con le seguenti caratteristiche.

a) Lunghezza finita

b) esiste un agente di calcolo capace di eseguire le istruzioni

c) l'agente ha a disposizione una memoria

(per immagazzinare i risultati intermedi del calcolo, da uscire nei passi successivi)

d) il calcolo procede per passi discreti

(non si basa su dispositivi analogici)

"calcolatore digitale"

e) il calcolo non è probabilistico

(non otteniamo un calcolo deterministico)

f) non dev'esserci alcun limite alla lunghezza dei dati di ingresso

g) non c'è alcun limite alla memoria disponibile

vogliamo definire algoritmi che funzionano per ogni possibile input!

(es: somma ... per addendi di ogni dimensione)

potrebbe sembrare memo. maturo, ma una memoria illimitata è indispensabile per la ragione di cui sopra, dato che per alcune funzioni lo spazio richiesto per i risultati intermedi dipende dall'input

$$\text{es: } f(n) = n^2$$

$$(1000)^2 = 1000\ 000$$

? devo aggiungere un numero
 di zeri che dip. da n

~ n deve essere memorizzabile
 (gli stati sono finiti)

h) deve esistere un limite finito $\left\{ \begin{array}{l} \text{al numero} \\ \text{alla complessità} \end{array} \right\}$ delle istruzioni

3

vuole catturare l'insieme finito di dispositivi di calcolo

(motivato da Turing con la limitatezza della mente / memoria umana.)

i) per un calcolatore la quantità di memoria indirizzabile con una singola istruzione dev'essere finita

(anche se nel suo complesso, per (g), la memoria è illimitata)

j) sono ammesse computazioni che terminano (e restituiscono un risultato) dopo un numero finito, ma illimitato di passi

(es: il quadrato richiede un numero di passi proporzionale all'argomento)

h) la computazione può non terminare

(e in questo caso non restituisce un risultato)

Prima di introdurre un modello di calcolo concreto, sul quale ci focalizzeremo e che ci permetterà di dare una def. completamente formale di funzioni calcolabili è interessante osservare che già queste semplici intuizioni ci permettono di concludere l'esistenza di funzioni non calcolabili.

* Esistenza di funzioni non calcolabili

Per essere formali... un po' di notazioni:

- \mathbb{N} insieme dei numeri naturali

$$\mathbb{N} = \{0, 1, 2, \dots\}$$

- prodotto cartesiano: A, B insiemi

$$A \times B = \{(a, b) \mid a \in A, b \in B\} = \{\{a, b\} \mid a \in A, b \in B\}$$

$$A^m = A \times \underbrace{\dots \times A}_{m \text{ volte}}$$

quindi

$$A^1 = A$$

$$A^{m+1} = A \times A^m$$

- relazioni / predici cat.

$$r \subseteq A \times B$$

- funzioni (parziali): $f: A \rightarrow B$

$$f \subseteq A \times B \quad \text{t.c. } \forall (a, b), (a, b') \in f \quad b = b'$$

$$\hookrightarrow \text{il dominio di } f \text{ è } \text{dom}(f) = \{a \in A \mid \exists b \in B \ (a, b) \in f\}$$

Sorivvediamo

$$f(a) \downarrow \quad f \text{ definita su } a \quad \text{per } a \in \text{dom}(f)$$

$$f(a) \uparrow \quad f \text{ indif. su } a \quad \therefore a \notin \text{dom}(f)$$

* Cardinalità

dato un insieme A indichiamo con $|A|$ la sua cardinalità
(intuit. il numero di elementi)

* dati A, B insiemi

$$|A| = |B| \quad \text{se esiste } f: A \rightarrow B \text{ bigettiva}$$

(intuitivamente hanno lo stesso num. di elementi)

$$|A| < |B| \quad \text{se esiste } f: A \rightarrow B \text{ iniettiva}$$

in particolare

$$\ast \text{ se } A \subseteq B \Rightarrow |A| \leq |B|$$

(come provato dalla funzione iniettiva
di inclusione)

$$i: A \rightarrow B$$

$$a \mapsto a$$

$$\ast \text{ se } |A| \leq |\mathbb{N}|$$

l'insieme A è detto numerabile

Sorgente del nome

esiste una funzione surgettiva $f: \mathbb{N} \rightarrow A$

per cui è possibile enumerare gli elementi di A uno di seguito all'altro

$$f(0) \ f(1) \ f(2)$$

$$a_0 \quad a_1 \quad a_2$$

$$\ast \text{ se } A \text{ e } B \text{ sono numerabili} \Rightarrow A \times B \text{ numerabile}$$

idea della prova

• essendo A e B numerabili posso considerare le enumerazioni

$$A \quad a_0 \ a_1 \ a_2$$

$$B \quad b_0 \ b_1 \ b_2$$

• gli elementi di $A \times B$ posso essere pensati in una matrice

$$b_0 \quad b_1 \quad b_2$$

$$a_0 \quad (a_0, b_0) \quad (a_0, b_1) \quad (a_0, b_2)$$

$$a_1 \quad (a_1, b_0) \quad (a_1, b_1) \quad (a_1, b_2)$$

$$a_2 \quad (a_2, b_0) \quad (a_2, b_1) \quad (a_2, b_2)$$

possono essere

enumerati come indicato

$$\ast \text{ se } \{A_i\}_{i \in \mathbb{N}} \text{ sono insiem numerabili} \Rightarrow \bigcup_{i \in \mathbb{N}} A_i \text{ numerabile}$$

9

Torniamo a noi... esistenza di funzioni non calcolabili

- consideriamo funzioni umane sui naturali $f: \mathbb{N} \rightarrow \mathbb{N}$
- indichiamo

$$F = \{f \mid f: \mathbb{N} \rightarrow \mathbb{N}\}$$

l'insieme di tutte le funzioni (potezionali)

- ricordiamo che una funzione è calcolabile se esiste un algoritmo che la calcola

es

per un certo modello di calcolo fissato, sia A l'insieme degli algoritmi

ogni algoritmo $A \in A$ calcola una funzione $f_A: \mathbb{N} \rightarrow \mathbb{N}$

- * sia $F_A = \{f_A \mid A \in A\}$. insieme delle funzioni (calcolabili nel modello di calcolo fissato)

cortoamente

$$F_A \subseteq F$$

↑ l'inclusione è stretta?

Basta osservare che

- un algoritmo $A \in A$ è una sequenza finita di istruzioni, prese da un insieme finito di istruzioni possibili I

?

$$A \subseteq \bigcup_{m \in \mathbb{N}} I^m$$

insieme numerabile di insiem numerabili!

$$|A| \leq |\bigcup_{m \in \mathbb{N}} I^m| \leq |\mathbb{N}|$$

è dato che la funzione

$$A \rightarrow F_A$$

$$A \rightarrow f_A$$

è surgettiva per dif

$$|F_A| \leq |A| \leq |\mathbb{N}|$$

Invece

T

$$|F_T| = |\mathbb{N} \rightarrow \mathbb{N}| \geq |\{f \mid f: \mathbb{N} \rightarrow \mathbb{N} \text{ totale}\}| > |\mathbb{N}|$$

Infatti, supponiamo per assurdo che T sia numerabile;

possiamo considerare

$f_0 \quad f_1 \quad f_2 \quad \dots$ enum di tutte le funz.

$$0 \quad f_0(0) \quad f_1(0) \quad f_2(0) \quad \dots$$

$$1 \quad f_0(1) \quad f_1(1) \quad f_2(1) \quad \dots$$

$$2 \quad f_0(2) \quad f_1(2) \quad f_2(2) \quad \dots$$

e costituire una funzione, che prendendo i valori sulla diagonale e modificandoli

$$d: \mathbb{N} \rightarrow \mathbb{N}$$

$$d(m) = f_m(m) + 1$$

- d totale

ASSURDO!

- $d \neq f_i \quad \forall i \in \mathbb{N}$

* Riassumendo

$$F_A \subseteq F$$

$$|F_A| \leq |\mathbb{N}| < |F|$$

purtanto

$$F_A \not\subseteq F$$

∴

• nessun formalismo calcola tutte le funzioni.

• le funzioni non calcolabili sono la maggioranza

* Quale modello di calcolo?

Si pone il problema: quale modello di calcolo utilizzare per formalizzare la nozione di algoritmo e di funzione calcolabile?

3

sono stati proposti vari modelli:

- macchina di Turing (Turing, 1936)
- λ-calcolo (Church, 1936)
- funzioni parziali ricorsive (Gödel - Kleene 1930)
- sistemi deduttivi combinatorici (Post, 1945)
- sistemi di Markov (Markov, 1951)
- Marchine a Registri illimitati (URM) (Sheperdson - Sturgis, 1963)

3

ogni modello di calcolo determina una classe di funzioni computabili

ma
3

per tutti i modelli citati la classe delle funzioni calcolabili è la stessa !!

Questo porta allo

Tesi di Church - Turing:

Una funzione è calcolabile mediante un procedimento effettivo (calcolabile in un qualunque modulo di calcolo finitistico, ovvero che rispetti le condizioni (a)-(l))

sse

è calcolabile da una macchina di Turing

Nota

- non è un teorema

data la sua natura informale, il riferimento al concetto di procedimento effettivo, non è dimostrabile

E qualcuno, es. Gurevich, sostiene che dovrebbe essere dimostrato sullo base di una ossiomorfe teoria formale delle cond. (a)-(P)

- molhissime proposte impiemtenti di modello di calcolo (tutte quelle emerse finora) confermano la tesi

A volte ci oppelleremo alla tesi di Church per acciuffare le dimostraz. del fatto che qualcosa è calcolabile

{

si può usare solo quando "non serve", quando può essere sostituita da una prova formale (che magari mostrerebbe l'idea sotto un cumulo di dettagli tecnici)

* Unlimited Register Machine (URM)

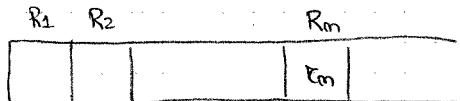
Come modello di calcolo per formalizzare le nozioni di funzione calcolab. utilizziamo la URM.

idealizzazione di un computer che esegue un linguaggio di tipo "goto" (~ assembler avocel!)

E' costituita da:

- memoria illimitata

che consiste in una sequenza infinita di registri ognuno dei quali può contenere un numero naturale (qualsiasi).



↑ il registro m^{mo} si indica con R_m .

il suo contenuto con x_m

la sequenza $x_1, x_2, \dots, x_m, \dots \in \mathbb{N}^\omega$ è detta configurazione della URM.

- agente computante

in grado di eseguire un programma URM

- programma URM

sequenza finita di istruzioni

J_1

J_2

J_s

che possono modificare "localmente" la configurazione della URM.

Tipi di Istruzioni

* Zero

$Z(m)$ azzera il contenuto del registro R_m
 $R_m \leftarrow 0$

* Successore

$S(m)$ incrementa il contenuto del registro R_m
 $R_m \leftarrow R_m + 1$

* Trasferimento

$T(m, m')$ trasferisce il contenuto del registro R_m nel registro $R_{m'}$
 R_m resta intatto
 $R_{m'} \leftarrow R_m$ (assegnamento)

Queste sono dette istruzioni aritmetiche, l'istruzione da eseguire al passo successivo è quella che segue l'istruzione corrente nel programma.

* Salto condizionale

$J(m, m', t)$ confronta il contenuto dei registri R_m e $R_{m'}$.

- se $R_m = R_{m'}$ salta all'istruzione t -ma
- altrimenti, continua con l'istruzione successiva

Computazione

Una computazione di una macchina VRM consiste nell'esecuzione di un programma P

$$P = I_1 - I_S$$

a partire da una configurazione iniziale dei registri

$$r_1 \ r_2 \ \dots \ r_m$$

• si inizia eseguendo la prima istruzione I_1 di P

• ad ogni passo la prossima istruzione da eseguire è determinata come indicato sopra.

Se la prossima istruzione da eseguire, non ci è (numero d'ordine > 5) la computazione ferma in una configuraz finale.

Questo può accadere se

- si esegue una istruzione S, Z, T o J con cond. falsa ed è l'ultima istruzione del programma.
- si esegue un'istruzione $J(m, m, t)$, la condizione è vera e $t > S$

Esempio

- I₁ J(2,3,5)
 I₂ S(1)
 I₃ S(3)
 I₄ J(1,1,1) ↪ salto incondizionato

Sembra preoccuparsi di cosa faccia questo programma vediamo una computazione a partire da

R ₁	R ₂	R ₃	
1	2	0	--



2	2	0	--
---	---	---	----



2	2	1	--
---	---	---	----



3	2	1	--
---	---	---	----



3	2	2	
---	---	---	--

STOP!

Cosa fa il programma?

* Semantica operazionale

più formalmente lo stato della macchina URM mentre esegue un programma $P = I_1 \dots I_s$ è dato da

$\langle \tau, t \rangle$

configurazione

$\tau: IN \rightarrow IN$ totale

$\tau(m) = \text{cont. del registro } R_m$

istruzione corrente (program counter)

si potrebbe facilmente definire una semantica operazionale

$\langle \tau, t \rangle \rightarrow \langle \tau', t' \rangle$

(non abbiamo bisogno di questo livello di formalità)

Notazione!

* Osservazione: Una computazione può non fermarsi!

$I_1 \quad S(1)$

1	0
---	---

$I_2 \quad J(1,1,1)$

2	0
---	---

3	0
---	---

Notazione: Sia P un programma URM

$a_1, a_2, a_3, \dots \in \mathbb{N}^w$ una sequenza di numeri naturali

indichiamo con $P(a_1, a_2, \dots)$ la computazione di P

a partire dalla configurazione iniziale $a_1 \mid a_2 \mid a_3 \mid \dots$

e

* $P(a_1, a_2, \dots) \downarrow$ se la computazione ferma

* $P(a_1, a_2, \dots) \uparrow$ se la computazione non ferma (o diverge)

Spesso (non sempre, per ovvie ragioni di finitezza dell'input) si considerano computazioni che partono da una configurazione iniziale nella quale solo un numero finito di registri contiene un valore $\neq 0$

Notazione: dati $a_1, a_2, \dots, a_k \in \mathbb{N}$

$P(a_1, \dots, a_k)$ denota la computazione $P(a_1, \dots, a_k, 0, \dots, 0)$

si estende a $P(a_1, \dots, a_k) \downarrow$

$P(a_1, \dots, a_k) \uparrow$

* Funzione URM-calcolabile

Sia $f: \mathbb{N}^k \rightarrow \mathbb{N}$ funzione parziale... cosa vuol dire che f è calcolabile da una URM?

Intuitivamente se esiste un programma P t.c. per ogni $(a_1, \dots, a_k) \in \mathbb{N}^k$ $P(a_1, \dots, a_k)$ calcola il valore di f

pb dove si mette l'output? convenzionalmente nel primo registro (gli altri per noi sono garbage...)

z

Notazione: sia P programma, $(a_1, \dots, a_k) \in \mathbb{N}^k$, scriviamo:

$P(a_1, \dots, a_k) \downarrow a$ se $P(a_1, \dots, a_k) \downarrow$

e la configuz. finale contiene a in R_1

Def (funzione URM-computabile)

Una funzione $f: \mathbb{N}^k \rightarrow \mathbb{N}$ si dice URM-computabile se esiste un programma URM P t.c.

$\forall (a_1, \dots, a_k) \in \mathbb{N}^k, a \in \mathbb{N}$

$P(a_1, \dots, a_k) \downarrow a$ se $(a_1, \dots, a_k) \in \text{dom}(f)$

e $f(a_1, \dots, a_k) = a$

In questo caso si dice che P calcola f

* si indica con

C classe delle funzioni URM-computabili

$e^{(k)}$

K-arie

quindi

$$e = \bigcup_{k \geq 1} e^{(k)}$$

* Esempi di funzioni computabili

1) $f: \mathbb{N}^2 \rightarrow \mathbb{N}$

$$f(x, y) = x + y$$

$J_1 \quad j(2, 3, 5)$

$J_2 \quad s(1)$

$J_3 \quad s(3)$

$J_4 \quad f(1, 1, 1)$

? salto imcondizionato

R_1	R_2	R_3	
x	y	0	--

Idea: inseriscono R_1 e R_3

fino a che R_2 e R_3

contengono la stessa valore

a questo punto ho aggiunto
y sia a R_1 che a R_3

? R_1 contiene $x+y$

2) $f: \mathbb{N} \rightarrow \mathbb{N}$

$$f(x) = x - 1 = \begin{cases} 0 & x=0 \\ x-1 & x>0 \end{cases}$$

idea

R_1	R_2	R_3	
x	0	0	

- se $x=0$ ok fine

- se $x>0$ tieni in R_2 K-1

$R_3 \leftarrow K$

com $K>1$ discende finché $R_3=x$

? a quel punto $R_2 = x-1$

ecce il programma.

$$I_1 \quad J(1, 4, 8)$$

$$I_2 \quad S(3)$$

$$I_3 \quad J(1, 3, 7)$$

$$I_4 \quad S(2)$$

$$I_5 \quad S(3)$$

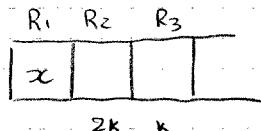
$$I_6 \quad J(1, 1, 3)$$

$$I_7 \quad T(2, 1)$$

3) $f: \mathbb{N} \rightarrow \mathbb{N}$

$$f(x) = \begin{cases} \frac{1}{2}x & x \text{ pari} \\ \uparrow & x \text{ dispari} \end{cases}$$

Idea:



$$I_1 \quad J(1, 2, 6)$$

$$I_2 \quad S(2)$$

$$I_3 \quad S(2)$$

$$I_4 \quad S(3)$$

$$I_5 \quad J(1, 1, 1)$$

$$I_6 \quad T(3, 1)$$

* Funzione calcolata da un programma

Dato un programma P e fissati $n \geq 1$ (numero di argomenti) è chiaro che esiste una unica funzione computata da P che (noi chiamiamo come)

$$f_P^{(n)} : \mathbb{N}^n \rightarrow \mathbb{N}$$

definita da

$$f_P^{(n)}(a_1, \dots, a_n) = \begin{cases} \text{l'unico } a \text{ t.c. } P(a_1, \dots, a_n) \downarrow a \\ \text{se } P(a_1, \dots, a_n) \downarrow \\ \uparrow \text{altrimenti (se } P(a_1, \dots, a_n) \uparrow) \end{cases}$$

Nota: Una stessa funzione può essere computata da più programmi diversi, per due motivi:

- possiamo aggiungere istruzioni inutili (non eseguibili, $T(m,m)$, ...)
- esistono algoritmi sostanzialmente diversi per la stessa funzione (es: ordinamento \sim quick sort, mergesort, ecc...)

3

ci sono infiniti programmi che calcolano la stessa funzione (o messo!)

Esercizio: Sia C' la classe delle funzioni calcolabili con programmi che non contengono l'istruzione di trasferimento T.

Allora, $C = C'$

dim

$$\ast \quad C \subseteq C'$$

ovvio

$$\ast \quad C \subseteq C'$$

Intuitivamente una istruzione di trasferimento

$$I_t \quad T(m, m)$$

può essere sostituita con una subroutine

$$\left. \begin{array}{l} I_t : J(m, m, t+1) \quad \leftarrow \text{caso } m=m \\ I_{t+1} : Z(m) \\ I_{t+2} : J(m, m, t+1) \\ I_{t+3} : S(m) \\ I_{t+4} : J(1, 1, t'+2) \end{array} \right\} \text{al posto giusto}$$

più precisamente.

Sia $f \in C \rightsquigarrow$ esiste un programma URM P , $k \in \mathbb{N}$ tale che

$$f = f_P^{(k)}$$

Dimostriamo per induzione sul numero h di istruzioni T in P che possiamo trasformare P in P'

- senza istruzioni T

$$- \text{t.c. } f_{P'}^{(k)} = f_P^{(k)}$$

$$\text{Mo } f = f_P^{(k)} = f_{P'}^{(k)} \in C'$$

($h=0$) Ovvio, basta prendere $P = P'$

($h \rightarrow h+1$) Sia P con $h+1$ istruzioni T

$$P : \quad I_1$$

:

$$I_h \quad T(m, m)$$

:

$$I_s$$

Si moti che si può assumere che il massimo numero d'ordine di istruzioni usato in P (in istruzioni $J(m, m, t)$) sia $s+1$ di modo che il programma termina sempre in $s+1$
 [se così non fosse basta sostituire $J(m, m, t)$ per $s+1$ con $J(m, m, s+1)$]

Quindi trasformiamo P in P''

I_1

:

I_6

$J(1, 1, s+2)$

P''

← substitutione in $s+2$

I_5

I_{s+1}

$J(1, 1, s+6)$

← salta alla fine

I_{s+2}

$J(m, m, t+1)$

I_{s+3}

$Z(m)$

I_{s+4}

$J(m, m, t+1)$

I_{s+5}

$S(m)$

I_{s+6}

$J(1, 1, s+4)$



Si ottiene così, P'' con le istruzioni $T(-, -)$ t.c.

$$f_{P''}^{(k)} = f_P^{(k)}$$

Per ipotesi induttiva si può dunque trasformare P'' in P' senza istruzioni T e tali che

$$f_{P'}^{(k)} = f_{P''}^{(k)} (= f_P^{(k)})$$

come desiderato.

Esercizio: Sia T_s l'istruzione di scambio, definita come segue

$T_s(m, m)$ scambia il contenuto dei registri R_m e R_m .

e sia URM_s la macchina URM dove ho l'istruzione T_s invece che T .
Detto C_s l'insieme delle funzioni URM_s -calcolabili

$$C_s = C$$

diam.

$$\ast \underline{C_s \subseteq C}$$

sia $f \in C_s \Rightarrow$ esiste un programma URM_s P , $k \in \mathbb{N}$ tc.
 $f_P^{(k)} = f$

vogliamo dimostrare che P può essere trasformato in un programma
URM P' che calcola la stessa funzione $f_{P'}^{(k)} = f_P^{(k)}$

Idea: codifico l'istruzione $T_s(m, m)$ con una routine URM

$T(m, i)$

$T(m, m)$

$T(i, m)$

dove i è un registro mom
utilizzato in P

Più formalmente; dimostriamo che:

per ogni programma P che contiene istruzioni $T(m, m)$ e $T_s(m, m)$
possiamo costruire P' , programma URM t.c.

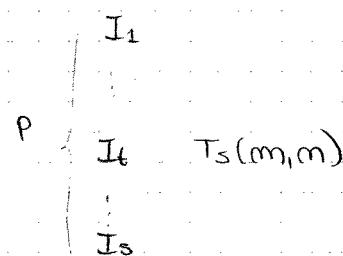
$$f_P^{(k)} = f_{P'}^{(k)}$$

↑
indispensabile: dopo un
passo di trasformazione ho
entrambe

Induzione sul numero h di istruz. T_s in P

($h=0$) ovvio, P è già un programma URM

($h \rightarrow h+1$) sia P con $h+1$ occorrenze di T_s .



Per codificare T_s mediante una "routine" prendiamo

i indice di un registro non utilizzato

$$i = \left(\max \{ \# \text{registro da comporre in } P \} \cup \{k\} \right) + 1$$

(parametro)

Inoltre, per sapere dove mettere la routine, conviene assumere che il massimo numero d'ordine di istruzioni usato in P (in istruzione $J(\dots)$) sia $s+1$ ma P termina sempre in $s+1$ no mettiamo la routine in $s+2$
così possiamo costruire P''

I₁

I₂ J(1,1,s+2)

I_s

I_{s+1} J(1,1,s+5) → molto allo fine!

I_{s+2} T(m,i)

I_{s+3} T(m,m)

I_{s+4} T(i,m)

I_{s+5} J(1,1,t+1)

} routine

E' facile convincersi che $f_{P''}^{(k)} = f_P^{(k)}$

Ora P'' contiene le occorrenze di T_s → per ipotesi \exists un programma VJM P' t.c.

$$f_{P'}^{(k)} = f_{P''}^{(k)} = f_P^{(k)}$$

Come desiderato.

* $e \subseteq e_s$

si può fare allo stesso modo, o più semplicemente usare l'esercizio preced.

$e \subseteq e' \subseteq e_s$

↑

sempre?

Esercizio

Dimostrare che le funzioni umarie URM-calcolabili senza istruzione J sono tutte del tipo

$$\lambda x. m$$

$$\lambda x. \lambda x + m$$

$$m \in \mathbb{N}$$

dim:

1° tentativo: sia P un programma senza istruzione J e dim. che $f_P^{(x)}$ è del tipo desiderato per $\ell(P)$, lunghezza di P

$$[\ell(P) = 1]$$

vari sotto casi a seconda di quale sia l'istruzione

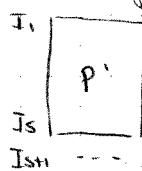
$$Z(m) \rightsquigarrow \begin{cases} m=1 & \lambda x. 0 \\ m \neq 1 & \lambda x. x \end{cases}$$

$$S(m) \rightsquigarrow \begin{cases} m=1 & \lambda x. x+1 \\ m > 1 & \lambda x. x \end{cases}$$

$$T(m,m) \rightsquigarrow \begin{cases} m=1 \wedge m \neq 1 & \lambda x. 0 \\ \text{altrimenti} & \lambda x. x \end{cases}$$

$$[\ell(P) = 5+1]$$

quindi il programma P sarà



per ipotesi induttiva P' calcola una funzione f' del tipo desiderato.

2 distinguiamo vari sotto casi a seconda del tipo di I_{St}

$$Z(m) \rightsquigarrow \begin{cases} m=1 & \lambda x. 0 \\ m > 1 & f' \end{cases}$$

$$S(m) \rightsquigarrow \begin{cases} m=1 & \lambda x. f'(x)+1 \\ m > 1 & f' \end{cases}$$

ok!
ok!

26)

$$T(m, m) \rightsquigarrow m \neq 1, m = 1$$

?

non posso dire niente!!

non so nulla di Rm !

3

situazione tipica: nel posso induzione della prova di $Q(-)$

$$Q(m) \Rightarrow Q(m+1) \text{ è falso}$$

questo non significa che sia falso lo statement

3

si può tentare di rafforzare la proprietà dimostrata e quindi l'ipotesi induttiva..

è un gioco di equilibrio: si parte da una ipotesi più forte

ma si deve dimostrare una tesi più forte!
(contrario a me)

Idea: proviamo a dimostrare che: indicata con $f_P^{(1)(K)}$ la funz.
calcolata da P con output nel registro K

per ogni programma P privo di istruzioni $J(-,-,-)$ e per ogni K

$$f_P^{(1)(K)} \text{ è del tipo } \begin{cases} \lambda x. m & m \in \mathbb{N} \\ \lambda x. x+m \end{cases}$$

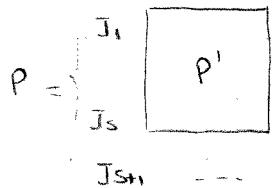
induzione su $\ell(P)$ [$\ell(P)=1$] sottocasi a seconda del tipo dell'istruzione

$$Z(m) \rightsquigarrow \begin{cases} m = K \vee K \neq 1 & \lambda x. 0 \\ m \neq K \wedge K = 1 & \lambda x. x \end{cases}$$

$$S(m) \rightsquigarrow \begin{cases} m = K \sim K = 1 & \lambda x. x+1 \\ K \neq 1 & \lambda x. 1 \\ m \neq K \sim K = 1 & \lambda x. x \\ K \neq 1 & \lambda x. 0 \end{cases}$$

$$T(m, m) \rightsquigarrow \begin{cases} m = K \sim m = 1 & \lambda x. x \\ m \neq 1 & \lambda x. 0 \\ m \neq K \sim K = 1 & \lambda x. x \end{cases}$$

[$\ell(p) = s+1$] quindi come nel caso precedente



con $f_{p'}^{(1)(k)}$ del tipo desiderato
per ogni k

distinguiamo vari sotto casi a seconda del tipo dell'istruzione I_{st+1}

$$Z(m) \sim \begin{cases} m = k & \lambda x. 0 \\ m \neq k & f_{p'}^{(1)(k)} \end{cases}$$

$$S(m) \sim \begin{cases} m = k & \lambda x. f_{p'}^{(1)(k)}(x) + 1 \\ m \neq k & f_{p'}^{(1)(k)} \end{cases} \quad \text{ok!}$$

$$T(m, m) \sim \begin{cases} m = k & f_{p'}^{(1)(m)} \\ m \neq k & f_{p'}^{(2)(k)} \end{cases}$$

N.B. In realtà mancano

A K A P ...

* Predicati decidibili

In matematica capita spesso di voler stabilire delle proprietà, ad esempio per $x, y \in \mathbb{N}$ possiamo considerare la proprietà

$$\text{div}(x, y) = "x \text{ è un divisore di } y"$$

ma non è altro che una relazione binaria

$$\text{div} \subseteq \mathbb{N} \times \mathbb{N}$$

Nell'ambito della teoria della calcolabilità si parla di predicati o problemi

* Un predicato k-ario su \mathbb{N} indicato com. $Q(x_1, \dots, x_k)$ è dunque una proprietà che può essere vera o falsa.

Formalmente possiamo vederlo come

- una funzione $Q: \mathbb{N}^k \rightarrow \{\text{true}, \text{false}\}$

- un insieme $Q \subseteq \mathbb{N}^k$

si scrive $Q(x_1, \dots, x_k)$ per indicare $(x_1, \dots, x_k) \in Q$

$$(o \ Q(x_1, \dots, x_k) = \text{true})$$

Quando è calcolabile? Int. quando esiste una URM che preso in input una k-pla x_1, \dots, x_k restituisce vero se $Q(x_1, \dots, x_k)$ è falso altremeni

come rappresentare vero/falso \Rightarrow convenzionalmente con 1/0

* Def (Predicato decidibile)

Un predicato $Q \subseteq \mathbb{N}^k$ si dice decidibile se la sua funzione caratteristica

$$\chi_Q(x_1, \dots, x_n) = \begin{cases} 1 & \text{se } Q(x_1, \dots, x_n) \\ 0 & \text{altrimenti} \end{cases}$$

è calcolabile.

Nota: χ_Q è una funzione totale (quando si studia la decidibilità di predicati si lavora solo con funzioni totali.)

E.s. 1: $E_g(x, y) = x = y$ è decidibile

infatti la funzione caratteristica $\chi_{E_g}(x, y) = \begin{cases} 1 & \text{se } x = y \\ 0 & \text{altrimenti} \end{cases}$

è calcolata da

$J(1,2, \text{si})$

NO: $Z(1)$

$J(1,1, \text{END})$

SI: $Z(1)$

$S(1)$

END

{ diametra poetica ...

Es 2: $Q(x) = x \text{ pari}$

LOOP: $T(1,2, \text{si})$

$S(2)$

$J(1,2, \text{No})$

$S(2)$

$J(1,1, \text{LOOP})$

SI: $S(3)$

NO: $T(3,1)$

$J(1,2, \text{si})$

$J(1,1, \text{No})$

oppure

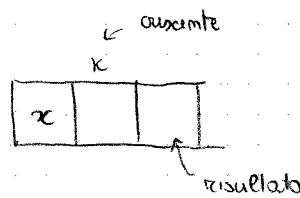
SI: $S(3)$

NO: $T(3,1)$

risultato in R_3

(che inizialmente
voleva 0)

buco!)



Es 3: $Q(x,y) = x \leq y$

Due possibilità:

a) incremento sia x che y

fimché $x+k = y \Rightarrow x \leq y$

oppure $y+k = x \Rightarrow x > y$

non = pur l'ordine
dei confronti

$T(1,3)$

$T(2,4)$

LOOP: $J(2,3, \text{si})$

$x+k = y ?$

$J(1,4, \text{No})$

$y+k = x ?$

$S(3)$

$S(4)$

$J(1,1, \text{LOOP})$

SI: $S(5)$

NO: $T(5,1)$

1	2	3	4	5	...
x	y	$x+k$	$y+k$...

risultato

b) incremento um registro a partire da 0 se raggiungo prima x

$\Rightarrow x < y$

altrimenti $x > y$

Loop: $J(1, 3, \text{si})$

$J(2, 3, \text{No})$

$S(3)$

$J(1, \text{loop})$

si: $S(4)$

No: $T(4, 1)$

$x+k$	k	risultato
x	y	
1	2	
3	4	

Esempio: predicato $\text{div}(x, y)$

[si doppia $x \neq 0$]

Loop: $J(2, 3, \text{si})$

$Z(4)$

/* somma x a R_2 */

ADDX: $J(1, 4, \text{loop})$

$J(2, 3, \text{No})$

/* se per $h < x$

$Kx+h=y$

$S(3)$

$\rightarrow \text{No! } x/$

$S(4)$

$J(1, 1, \text{ADDX})$

x	y	h	$Kx+h$
1	2		
3	4		

si: $S(5)$

No: $T(5, 1)$

* Computabilità su altri domini

Dato che la UTM manipola solo numeri naturali, la nostra definizione di calcolabilità riguarda solo funzioni e predicationi su \mathbb{N}

Il concetto di calcolabilità può essere esteso ad altri domini ... appellaindosi ad una mazzone (un po' imprecisa...) di codifica effettiva

Sia D un dominio di oggetti presentati in modo effettivo ovvero tale che esiste una funzione di codifica

$$\alpha : D \rightarrow \mathbb{N}$$

- biumivoca (basta in realtà immettere, ma non muore)
- effettiva

? non possiamo avere una mazzone formale di effettività ... si intende che tutti sarebbero d'accordo sulla sua calcolabilità (se al mondo c'è giustizia...)

Può essere $D = \mathbb{Q}$

$D \subseteq A^*$ per un alfabeto A

$$D = \mathbb{Z}$$

$D = \mathbb{N}^*$ liste di interi

ma non $D = \mathbb{R}$ o $D = A^\omega$ (stream)

ξ

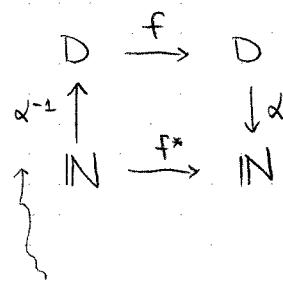
Una funzione $D \rightarrow D$ si dice calcolabile se lo è la sua "codifica" tramite α

Def: Una funzione $f: D \rightarrow D$ è calcolabile

se è calcolabile la funzione

$$f^*: \mathbb{N} \rightarrow \mathbb{N}$$

$$f^* = \alpha \circ f \circ \alpha^{-1}$$



vedremo che se α è effettivo, lo è anche la sua inversa

Note: Permane il pb relativo all'informalità della mazzone di effettività,

Computabilità - 2020/2021 - LM Informatica - Università di Padova questo il resto è formale.

Esempio - Per $D = \mathbb{Z}$ possiamo considerare la codifica effettiva

$$\alpha: \mathbb{Z} \rightarrow \mathbb{N}$$

$$\alpha(z) = \begin{cases} 2z & z \geq 0 \\ -2z-1 & z < 0 \end{cases} \quad \text{effettiva}$$

con inversa

$$\alpha^{-1}: \mathbb{N} \rightarrow \mathbb{Z}$$

$$\alpha^{-1}(m) = \begin{cases} \frac{1}{2}m & m \text{ pari} \\ -\frac{1}{2}(m+1) & m \text{ dispari} \end{cases}$$

La funzione

$$f: \mathbb{Z} \rightarrow \mathbb{Z} \quad \text{è URM-calcolabile}$$

$$z \mapsto |z|$$

Impath

$$f^*: \mathbb{N} \rightarrow \mathbb{N} \quad \text{m. pari} \quad \alpha(f(\frac{1}{2}m)) = \alpha\left(\frac{1}{2}m\right) = m$$

$$f^*(m) = \alpha \cdot f \cdot \alpha^{-1}(m) = \begin{cases} m & m \text{ pari} \\ m+1 & m \text{ dispari} \end{cases} \quad \alpha(f(-\frac{1}{2}(m+1))) = \alpha\left(\frac{1}{2}(m+1)\right) = m+1$$

ovvero

$$f^*(m) = \begin{cases} m & m \text{ pari} \\ m+1 & m \text{ dispari} \end{cases}$$

URM-calcolabile!

Generazione di funzioni calcolabili

Analizziamo delle tecniche per "combinare" funzioni calcolabili in modo da ottenere nuove funzioni che sono ancora calcolabili.
Più precisamente dimostreremo che la classe \mathcal{C} è chiusa rispetto alle operazioni di:

- composizione (generalizzata)
- ricorsione primitiva
- minimizzazione (illimitata)

Così, per dimostrare che una funzione $f: \text{IN}^k \rightarrow \text{IN}$ è calcolabile avremo a disposizione 2 modi:

- scrivere il programma URTI P che calcola f ($f_P^{(k)} = f$)
- sfruttare i teoremi di chiusura di \mathcal{C}

In realtà, le tre operazioni considerate non sono scelte a caso

obiettivo di lungo termine: mostrare che \mathcal{C} coincide con la classe di funzioni generalizzabili mediante composiz., ricorsione primitiva e minimizzazione, a partire da un nucleo ristretto di funzioni di base (funzioni parziali ricorsive di Gödel - Kleene)

Notazione: Date $f, g: \text{IN}^k \rightarrow \text{IN}$ e $\vec{x} \in \text{IN}^k$ scriviamo

$$f(\vec{x}) \simeq g(\vec{x}) \quad \text{per} \quad \begin{aligned} f(\vec{x}) \downarrow &\iff g(\vec{x}) \downarrow \\ \text{e se } f(\vec{x}) \downarrow &\Rightarrow f(\vec{x}) = g(\vec{x}) \end{aligned}$$

* Funzioni di base

Le seguenti funzioni sono calcolabili.

1) zero $\underline{0}: \text{IN}^k \rightarrow \text{IN}$

$$\lambda x_1 \dots x_k. 0$$

2) successore $s: \text{IN} \rightarrow \text{IN}$

$$\lambda x. x+1$$

3) proiezione $U_i^k: \text{IN}^k \rightarrow \text{IN}$

$$\lambda x_1 \dots x_k. x_i$$

3A)

dim

i programmi che calcolano (1), (2), (3) sono le istruzioni (aritmetiche)

- 1) 0 calcolata da $\Xi(1)$
- 2) S " " $S(1)$
- 3) U_i^k " " $T(i, 1)$

□

Per dimostrare le proprietà di chiusura, avremo bisogno di "combinare" programmi... ci serve un po' di motivazione

* Dato un programma URM P

- $p(P)$ massimo numero di registro usato in P
- P è in forma standard se, detta s la sua lunghezza, per ogni istruzione $J(m, m, t)$ $t \leq s+1$
 (\rightsquigarrow se fermissima, fermiamo all'istruzione $s+1$)

E' ovvio che non è limitativo considerare solo programmi in forma standard
 ovvero vale

Lemma : Per ogni programma URM P esiste un programma P' in forma standard, equivalente ovvero f.c.

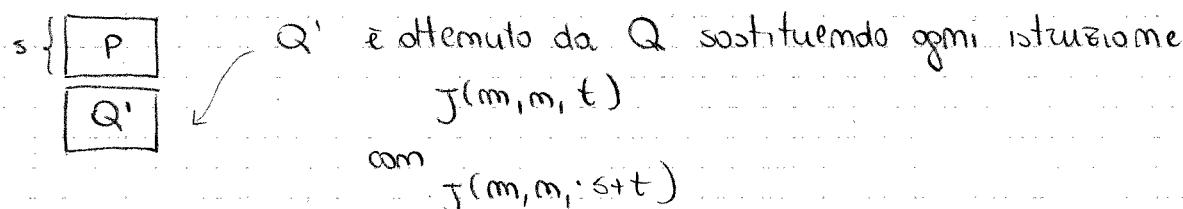
$$\forall k \quad f_P^{(k)} = f_{P'}^{(k)}$$

dim

basta sostituire ogni istruzione $J(m, m, t)$ in P f.c. $t > s+1$ con
 $J(m, m, s+1)$ □

* Concatenazione di programmi

Dati due programmi P, Q in forma standard, P di lungh. s , la loro concatenazione è



Nota :

- se P, Q im forma standard $\Rightarrow P \circ Q$ im forma standard
- $(P \circ Q) \circ R = P \circ (Q \circ R)$

3

assumiamo che ogni programma sia im forma standard e useremo la concatenazione "liberamente"

Convenzionalmente, dato un programma P pensiamo che calcoli una funzione prendendo l'input da R_1, \dots, R_k e mettendo l'output in R_i

3

ci sarà utile prendere l'input e fornire l'output in registri arbitrari

$$R_{i_1} \dots R_{i_k} \quad R_e$$

senza assumere che gli altri registri siano a 0 o a puoi

3
costruiamo

$$P[i_1, \dots, i_k \rightarrow e]$$

$$T(i_1, 1)$$

$$\vdots$$

$$T(i_k, k)$$

$$Z([k+1, p(P)]) \quad \text{licenza poch}$$

$$P$$

$$T(1, e)$$

Possiamo ora dimostrare le proprietà di chiusura...

* Composizione generalizzata

Mostriamo in primo luogo che la classe C è chiusa rispetto ad una forma generalizzata di composizione (int. composizione \Leftrightarrow concatenazione!)

Def. Siamo $f: IN^K \rightarrow IN$,

$$g_i: IN^m \rightarrow IN \quad i=1, \dots, K$$

la composizione di f, g_1, g_K è la funzione

$$h: IN^m \rightarrow IN$$

$$h(x_1, \dots, x_m) \approx f(g_1(x_1, \dots, x_m), \dots, g_K(x_1, \dots, x_m))$$

(resto inteso che $h(\bar{x}) \downarrow \Leftrightarrow g_i(\bar{x}) \downarrow \forall i \text{ e } f(g_1(\bar{x}), \dots, g_K(\bar{x})) \downarrow$)

36

Proposizione: Se $f: \mathbb{N}^k \rightarrow \mathbb{N}$, $g_i: \mathbb{N}^m \rightarrow \mathbb{N}$, $i=1, \dots, k$ sono calcolabili allora anche $h: \mathbb{N}^m \rightarrow \mathbb{N}$

$$h(x_1, \dots, x_m) = f(g_1(x_1, \dots, x_m), \dots, g_k(x_1, \dots, x_m))$$

è calcolabile.

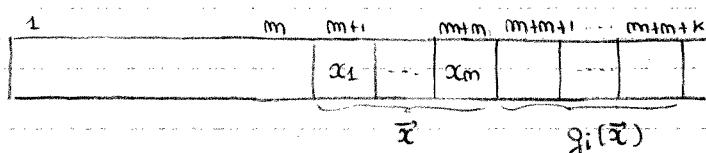
dimm

sia mo. F, G_1, \dots, G_k programmi in forma standard per f, g_1, \dots, g_k

consideriamo un numero di registro non usato da nessuno

$$m = \max \{ p(F), p(G_1), \dots, p(G_k), k, m \}$$

il programma per la composizione può essere



$$T([1, m], [m+1, m+m])$$

$$G_1 [m+1, \dots, m+m \rightarrow m+m+1]$$

:

$$G_k [m+m+1, \dots, m+m+k \rightarrow m+m+k]$$

$$P[m+m+k+1, \dots, m+m+k \rightarrow 1]$$

□

Corollario: Sia $f: \mathbb{N}^k \rightarrow \mathbb{N}$ calcolabile.

Allora

$$g: \mathbb{N}^m \rightarrow \mathbb{N}$$

$$g(x_1, \dots, x_m) = f(x_{i_1}, \dots, x_{i_k})$$

↓
seq. di variabili in x_1, \dots, x_m con ripetizioni

e assenze

è calcolabile.

dimm

$$\text{se } \vec{x} = (x_1, \dots, x_m)$$

$$g(\vec{x}) = f(U_{i_1}^m(\vec{x}), \dots, U_{i_k}^m(\vec{x}))$$

□

Esempi: Se $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ è calcolabile. Bo sono anche

$$f_1(x, y) = f(y, x)$$

$$f_2(x) = f(x, x)$$

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

Nota : Sulla base di questo risultato possiamo utilizzare la comp. generalizzata anche quando le gi. non sono funzioni di tutte le variabili o sono funzioni con ripetizioni.

Esempio: sapendo che

$$\text{sum}(x_1, x_2) = x_1 + x_2$$

è calcolabile possiamo dedurre che lo è anche

$$f(x_1, x_2, x_3) = x_1 + x_2 + x_3$$

infatti

$$f(x_1, x_2, x_3) = \text{sum}(\text{sum}(x_1, x_2), x_3)$$

passo per otte come funzioni $\mathbb{N}^3 \rightarrow \mathbb{N}$

$$\text{sum}(\text{sum}(U_1^3(\vec{x}), U_2^3(\vec{x})), U_3^3(\vec{x}))$$

Esempi : Sono calcolabili le seguenti funzioni.

* costante m $\lambda \vec{x}. m$

$$m(\vec{x}) = s(s(\dots s(0(\vec{x})))$$

* somma degli argomenti

$$g(x_1, \dots, x_k) = x_1 + \dots + x_k \quad (\text{vedi sopra})$$

* prodotto per una costante

$$m \cdot x = g(x, \dots, x) \quad \text{m volte}$$

dove g è la funz. al passo prec.

* se $f(x, y)$ calcolabile lo è anche

$$f'(x) = f(x, m)$$

infatti

$$f'(x) = f(x, m) = f(U_1^1(x), m(x))$$

* se $f: \mathbb{N} \rightarrow \mathbb{N}$ totale calcolabile

il predicato

$$Q(x, y) \equiv f(x) = y$$

è decidibile

infatti, sappiamo che $\chi_{E_q}(x, y) = \begin{cases} 1 & x = y \\ 0 & \text{altrimenti} \end{cases}$ è calcolabile

38) dunque

$$\begin{aligned} \chi_a(x,y) &= \chi_{Eq}(g(x),y) \\ &= \chi_{Eq}(g(U_1^2(x,y)), U_2^2(x,y)) \end{aligned}$$

Ricorsione Primitiva

Ricorsione: concetto familiare! Permette di definire una funzione specificandomi i valori in termini di altri valori della funzione stessa (e possibilmente utilizzando altre funzioni già definite).

* fattoriale $0! = 1$

$$(m+1)! = (m+1) \cdot m!$$

Si usa *

* fibonacci $f(0) = 1$

$$f(1) = 1$$

$$f(m+2) = f(m) + f(m+1)$$

ce me sono tornati in mente, qui usiamo una forma "comballata" di ricorsione.

Def (Ricorsione primitiva)

Date le funzioni

$$f: \mathbb{N}^k \rightarrow \mathbb{N}$$

$$g: \mathbb{N}^{k+2} \rightarrow \mathbb{N}$$

definiamo $h: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$

$$h(\vec{x}, 0) \simeq f(\vec{x})$$

$$h(\vec{x}, y+1) \simeq g(\vec{x}, y, h(\vec{x}, y))$$

Nota: La funzione h è definita in modo equazionale come

h che compare ad entrambi i lati: definizione implicita, non ovvia

- non è ovvio che una tale h esista!

- non è ovvio che, se esiste, sia unica!

In realtà esiste e unica, ma una teoria generale che supporti questa asserzione è meno banale...

L'argomento procede come segue:

- indichiamo con $[N^m \rightarrow N]$ l'insieme delle funzioni di m argomenti.

- definiamo un operatore

$$T : [N^{k+1} \rightarrow N] \rightarrow [N^{k+1} \rightarrow N]$$

$$T(h)(\vec{x}, 0) = f(\vec{x})$$

$$T(h)(\vec{x}, y+1) = g(\vec{x}, y, h(\vec{x}, y)) \quad \leftarrow \text{nessuna circolarità}$$

- le funzioni cercate sono i punti fissi di T

$$h \text{ t.c. } T(h) = h$$

- esistenza del punto fisso

- $[N^{k+1} \rightarrow N]$ cpo
- T continuo

$\} \xrightarrow{\text{scott}} \text{ha un minimo punto fisso}$

- unicità

induttivamente se h, h' punti fissi $\Rightarrow h = h'$

* Osservazione

Funzioni definite per

1 - composizione generalizzata

2 - ricorsione primitiva

a partire da funzioni totali sono ancora totali

dim

1 - ovvio dalla definizione

2 - siamo $f: IN^k \rightarrow N$, $g: IN^{k+2} \rightarrow N$ totali e definiamo

$$h: IN^{k+1} \rightarrow N$$

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

$$h(\vec{x}, y+1) = g(\vec{x}, y, h(\vec{x}, y))$$

si può provare per induzione su y che $\forall \bar{x} \quad (\bar{x}, y) \in \text{dom}(h)$

$$(y=0) \quad h(\bar{x}, 0) \simeq f(\bar{x}) \downarrow$$

$$(y \rightarrow y+1) \quad h(\bar{x}, y+1) \simeq g(\bar{x}, y, \underline{h(\bar{x}, y)}) \downarrow$$

$\downarrow \times \text{lp.ind.}$

□

Esempi:

* somma $x+y$

$$x+0 = x$$

$$x+(y+1) = x+y+1$$

∴

$$h(x,0) = x$$

$$h(x,y+1) = h(x,y)+1$$

$$f(x) = x$$

$$g(x,y,z) = z+1$$

* prodotto $x \cdot y$

$$x \cdot 0 = 0$$

$$x(y+1) = xy + x$$

∴

$$h(x,0) = 0$$

$$h(x,y+1) = h(x,y) + x$$

$$f(x) = 0$$

$$g(x,y,z) = z+x$$

* fattoriale $y!$

$$0! = 1$$

$$(y+1)! = y!(y+1)$$

$$h(0) = 1$$

$$h(y+1) = h(y)(y+1)$$

$$f = 1 \text{ costante!}$$

$$g(y,z) = z(y+1)$$

* Proposizione (chiarezza rispetto alla ricorsione primitiva di e)

Siamo $f: \mathbb{N}^k \rightarrow \mathbb{N}$ e $g: \mathbb{N}^{k+2} \rightarrow \mathbb{N}$ calcolabili

Allora $h: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ definito per ricorsione primitiva

$$h(\bar{x},0) = f(\bar{x})$$

$$h(\bar{x},y+1) = g(\bar{x},y, h(\bar{x},y))$$

è calcolabile

dim

Siamo F, G programmi in forma standard per f, g

mostriamo come si possa costruire un programma per h

42) si procede come suggerito dalla definizione

- si parte da

$\boxed{x_1 \dots x_k} \boxed{y \mid 0}$

- si salvano i parametri e si inizia a calcolare $h(\vec{x}, 0)$ usando F
se $y = 0$ abbiamo finito, altrimenti salvo $h(\vec{x}, 0)$ e

calcolo $h(\vec{x}, 1) = g(\vec{x}, 0, h(\vec{x}, 0))$ con G

$h(\vec{x}, i)$

fino ad arrivare a $i = y$

Al solito abbiamo bisogno di registri non usati da F, G

$$m = \max \{ K+2, p(F), p(G) \}$$

e costruiamo il programma per h come segue:

1	-	-	m+1	-	m+k	m+k+1	m+k+3
							$m+k+2$

argomenti di g

T ([1, k], [m+1, m+k])

/* copia \vec{x} */

T (K+1, m+k+3)

/* copia y */

F [m+1, ..., m+k → m+k+2]

/* $h(\vec{x}, 0)$ */

Loop: J (m+k+1, m+k+3, END)

/* $i = y ?$ */

G [m+1, ..., m+k+2 → m+k+2] /* $h(\vec{x}, i+1) = g(\vec{x}, i, h(\vec{x}, i))$ */

S(m+k+4)

/* $i = i + 1$ */

J (1, 1, Loop)

END: T (m+k+2, 1)

□

Nota: Non si fa altro che implementare la ricorsione con l'iterazione!

* Teorema

Le seguenti funzioni sono calcolabili:

1) Somma

$$x + y \quad (\text{vedi prec.})$$

2) Prodotto

$$x * y \quad (\text{vedi prec.})$$

3) Eseponenziale

$$x^y$$

$$\begin{cases} x^0 = 1 \\ x^{y+1} = x^y \cdot x \end{cases}$$

\rightsquigarrow

$$h(x, 0) = 1$$

$$f(z) = 1$$

$$h(x, y+1) = h(x, y) \cdot x$$

$$g(x, y, z) = z \cdot x$$

4) Predecessore

$$x - 1$$

$$0 - 1 = 0$$

$$h(0) = 0$$

$$f = 0$$

$$(x + 1) - 1 = x$$

$$h(x+1) = x$$

$$g(y, z) = y$$

$$5) x - y = \begin{cases} x - y & x \geq y \\ 0 & \text{altrimenti} \end{cases}$$

$$x - 0 = x$$

$$f(x) = x$$

$$x - (y+1) = (x - y) - 1$$

$$g(z, y, z) = z - 1$$

$$6) sg(x) = \begin{cases} 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

$$sg(0) = 0$$

$$f = 0$$

$$sg(x+1) = 1$$

$$g(y, z) = 1$$

$$7) \overline{sg}(x) = \begin{cases} 1 & x = 0 \\ 0 & x > 0 \end{cases}$$

$$\overline{sg}(x) = 1 - sg(x)$$

in composizione e (6)

} pesante, ma diamo costruendo
una "libreria" per la macchina
universale:
juke box dei programmi
esegui il pig. 1 - 5,

$$8) |x-y| = \begin{cases} x-y & x \geq y \\ y-x & x < y \end{cases}$$

$$|x-y| = (x-y) + (y-x) \quad \rightsquigarrow \text{de (1), (6) e composiz.}$$

9) fattoriale $y!$

$$0' = 1$$

$$(y+1)' = (y+1) \cdot y'$$

$$f \in 1$$

$$g(y, z) = (y+1) \cdot z$$

10) mim (x, y)

$$\text{mim}(x, y) = x - (x-y)$$

11) max (x, y)

$$\text{max}(x, y) = x - y + y$$

$$12) \text{rem}(x, y) = \begin{cases} y \bmod x & x \neq 0 \\ y & x = 0 \end{cases}$$

resto della divisione intera di y per x . (convenzione! ragionevole se il resto $\text{rem}(x, y)$ deve essere tali che $\exists k. k \cdot x + \text{rem}(x, y) = y$)

$$\text{rem}(x, 0) = 0$$

$$\text{rem}(x, y+1) = \begin{cases} \text{rem}(x, y) + 1 & \text{se } \text{rem}(x, y) + 1 \neq x \\ 0 & \text{altrimenti} \end{cases}$$

$$= (\text{rem}(x, y) + 1) \cdot \text{sg}((x-1) - \text{rem}(x, y))$$

$$\rightsquigarrow f(x) = 0 \quad g(x, y, z) = z * \text{sg}(x-1 - z) \quad \text{ok!}$$



$$13) \text{ qt}(x, y) = y \text{ div } x$$

(per convenzione: $\text{qt}(0, y) = y$)

difiniamo

$$\text{qt}(x, 0) = 0$$

$$\text{qt}(x, y+1) = \begin{cases} \text{qt}(x, y) + 1 & \text{se } \text{rm}(x, y) + 1 = x \\ \text{qt}(x, y) & \text{altrimenti} \end{cases}$$

$$= \text{qt}(x, y) + \text{sg}((x-1) - \text{rm}(x, y))$$

$$14) \text{ div}(x, y) = \begin{cases} 1 & \text{se } x \mid y \\ 0 & \text{altrimenti} \end{cases}$$

convenzione
0 | 0 & 0 | y & y > 0

$$\text{div}(x, y) = \overline{\text{sg}}(\text{rm}(x, y))$$

Come immediata applicazione abbiamo:

Corollario (Definizione per così)

Siamo $f_1, \dots, f_m : \mathbb{N}^k \rightarrow \mathbb{N}$ funzioni calcolabili totali

$Q_1, \dots, Q_m \subseteq \mathbb{N}^k$ predicati decidibili e mutuamente esclusivi
(per ogni $\vec{x} \in \mathbb{N}^k$ esattamente uno tra

$Q_1(\vec{x}), \dots, Q_m(\vec{x})$ vale)

allora $f : \mathbb{N}^k \rightarrow \mathbb{N}$ dif. da

$$f(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{se } Q_1(\vec{x}) \\ f_2(\vec{x}) & \text{se } Q_2(\vec{x}) \\ \vdots & \vdots \\ f_m(\vec{x}) & \text{se } Q_m(\vec{x}) \end{cases}$$

è calcolabile (totale)

dim

$$f(\vec{x}) = f_1(\vec{x}) \cdot \chi_{Q_1}(\vec{x}) + \dots + f_m(\vec{x}) \cdot \chi_{Q_m}(\vec{x})$$

si conclude, per composizione usando la calcolabilità di somma e prodotto.



Corollario (Algebra della decidibilità)

Siamo Q, Q' predici decidiibili. Allora sono decidiibili:

- 1) $\neg Q$
- 2) $Q \wedge Q'$
- 3) $Q \vee Q'$

dim.

Basta osservare che le funzioni caratteristiche

$$1) \chi_{\neg Q}(x) = 1 - \chi_Q(x)$$

$$2) \chi_{Q \wedge Q'}(x) = \chi_Q(x) \cdot \chi_{Q'}(x)$$

$$3) \text{ si osserva che } Q \vee Q' \equiv \neg(\neg Q \wedge \neg Q')$$

oppure

$$\chi_{Q \vee Q'}(x) = \max\{\chi_Q(x), \chi_{Q'}(x)\}$$

□

Ricordiamo che $\{\neg, \wedge, \vee\}$ (basta $\{\neg, \wedge\}$) è un insieme di connettivi funzionalmente completo (permette di esprimere ogni funzione $\{0,1\}^m \rightarrow \{0,1\}$) si deduce

Corollario Siamo $Q_1, \dots, Q_m \in \mathbb{N}^k$ predici decidiibili e sia

$f: \{0,1\}^m \rightarrow \{0,1\}$ una funzione. Si consideri

$$\chi: \mathbb{N}^k \rightarrow \{0,1\}$$

$$\chi(\vec{x}) = f(\chi_{Q_1}(\vec{x}), \dots, \chi_{Q_m}(\vec{x}))$$

Allora il predicibile Q corrisp. a χ è decidibile e quindi χ è calcolabile.

* Somme, prodotti, quanitificazioni limitate

Def. (Somma e prodotto limitato): sia $f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ totale.

* $\sum_{z < y} f(\vec{x}, z)$ è definito da

$$\sum_{z < 0} f(\vec{x}, z) = 0$$

$$\sum_{z < y+1} f(\vec{x}, z) = \sum_{z < y} f(\vec{x}, z) + f(\vec{x}, y)$$

* $\prod_{z < y} f(\vec{x}, z)$ è definito da

$$\prod_{z < 1} f(\vec{x}, z) = 1$$

$$\prod_{z < y+1} f(\vec{x}, z) = (\prod_{z < y} f(\vec{x}, z)) \cdot f(\vec{x}, y)$$

OSSERVAZIONE: se $f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ è calcolabile totale allora

1) $g(\vec{x}, y) = \sum_{z < y} f(\vec{x}, z)$

sono calcolabili totali

2) $h(\vec{x}, y) = \prod_{z < y} f(\vec{x}, z)$

distr

sono definite per ricorsione primitiva!

1) $g(\vec{x}, 0) = 0$

$$g(\vec{x}, y+1) = g(\vec{x}, y) + f(\vec{x}, y)$$

e +, f sono calcolabili

□

2) idem

Ovviamente, per composizione, il bound può essere una funzione totale calcolabile.

Altra conseguenza immediata riguarda la decidibilità delle quanitificazioni limitate su predicati

48)

Lemma: Sia $Q \subseteq \mathbb{N}^{k+1}$ un predicato decidibile. Allora

1. $Q_1(\bar{x}, y) = \forall z < y. Q(\bar{x}, z)$ sono decidibili
2. $Q_2(\bar{x}, y) = \exists z < y. Q(\bar{x}, z)$

dim

1. si osserva che $\chi_{Q_1}(\bar{x}, y) = \prod_{z < y} \chi_Q(\bar{x}, z)$.

2. $\chi_{Q_2}(\bar{x}, y) = \text{sg} \left(\sum_{z < y} \chi_Q(\bar{x}, z) \right)$ □

- Operazione di minimizzazione limitata

Data una funzione $f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ totale, definiamo $h: \mathbb{N}^{k+2} \rightarrow \mathbb{N}$ come segue

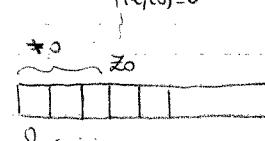
$$h(\bar{x}, y) = \mu z < y. f(\bar{x}, z) = \begin{cases} \text{min. } z < y \text{ t.c. } f(\bar{x}, z) = 0, & \text{se esiste} \\ y & \text{altrimenti} \end{cases}$$

Lemma: sia $f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ calcolabile totale. Allora anche $h: \mathbb{N}^k \rightarrow \mathbb{N}$ def. da $h(\bar{x}, y) = \mu z < y. f(\bar{x}, z)$ è calcolabile (totale)

dim

osserviamo che h può essere definita come

$$h(\bar{x}, y) = \sum_{z < y} \prod_{w < z} \text{sg}(f(\bar{x}, w))$$



vale i megli intervalli $[0, z]$ in cui $f \neq 0$, che se z_0

è il min $z < y$ dove f è nulla, sono proprio z_0

la somma esterna li conta.

alternativamente h può essere definita per ricorsione primitiva diretta.

$$\begin{aligned} h(\bar{x}, 0) &= 0 \\ h(\bar{x}, y+1) &= \begin{cases} \text{se } h(\bar{x}, y) \neq y & \rightsquigarrow h(\bar{x}, y) \\ \text{altrimenti} & \rightsquigarrow \begin{cases} \text{se } f(\bar{x}, y) = 0 & \rightsquigarrow y \\ \text{altrimenti} & \rightsquigarrow y+1 \end{cases} \end{cases} \\ &= \text{sg}(y - h(\bar{x}, y)) \cdot h(\bar{x}, y) + \overline{\text{sg}}(y - h(\bar{x}, y)) (y + \text{sg}(f(\bar{x}, y))) \end{aligned}$$

Nota: Anche qui il bound può essere una funzione totale calcolabile

Lemma: Le seguenti funzioni sono calcolabili:

$$a) D(x) = \text{numero di divisori di } x$$

$$b) P_r(x) = \begin{cases} 1 & \text{se } x \text{ primo} \\ 0 & \text{altrimenti} \end{cases} \quad (x \text{ primo decidibile})$$

$$c) p_x = x^{\text{mo numero primo}} \quad (\text{con } p_0 = 0 \rightsquigarrow p_1 = 2, p_2 = 3, \dots)$$

$$d) (x)_y = \begin{cases} \text{esponente di } p_y \text{ nella fattorizzazione di } x, x, y > 0 \\ 0 \quad \text{se } x=0 \text{ oppure } y=0 \end{cases}$$

$$\text{es: } 72 = 2^3 \cdot 3^2$$

$$(72)_1 = 3 \quad (72)_2 = 2 \quad (72)_3 = 0 \dots$$

dim

$$a) D(x) = \sum_{y \leq x} \text{div}(y, x)$$

$$b) P_r(x) = \begin{cases} 1 & \text{se } D(x) = 2 \\ 0 & \text{altrimenti} \end{cases} \quad \rightsquigarrow x > 1 \text{ ed è divisibile da 1 e se stesso}$$

$$= \overline{\text{sg}}(1|D(x)-2|)$$

$$c) p_x \text{ si può definire per ricorsione primitiva}$$

$$p_0 = 0$$

$$p_{x+1} = \mu z \leq (p_x! + 1) \cdot \overline{\text{sg}}(P_r(z) \cdot \chi_{z \geq p_x}(z))$$

Nota: certamente $p_{x+1} \leq p_x! + 1$ (e questo ci permette di fissare il bound!)

infatti detto p un primo nella decomposizione di $p_x! + 1$ (72) quindi

$$p \mid p_x! + 1$$

certamente $p > p_x$, altrettanto $p \mid p_x!$ e quindi $p \nmid 1$.

Pertanto $p_x < p_{x+1} \leq p$

d) si metti che

$$(x)_y = \max Z \text{ t.c. } p_y^z \mid x$$

$$= \min Z \text{ t.c. } p_y^{z+1} \nmid x = \mu z \leq x \cdot \text{div}((p_y)^{z+1}, x)$$

Esercizi (per cosa)

Dimostrare che sono computabili le seguenti funzioni

a) $\lfloor \sqrt{x} \rfloor$

$$\lfloor \sqrt{x} \rfloor = \max_{y \leq x} y^2 \leq x$$

$$= \min_{y \leq x} (y+1)^2 > x$$

$$= \mu_{y \leq x} ((x+1) - (y+1)^2)$$

b) $\text{mcm}(x,y)$

$$\text{mcm}(x,y) = \mu_{z \leq x \cdot y} (x|z \wedge y|z)$$

$$= \mu_{z \leq x \cdot y} \exists_{\overline{g}} (\text{div}(x,z) \wedge \text{div}(y,z))$$

c) $\text{MCD}(x,y)$

Certamente $\text{MCD}(x,y) \leq \min\{x,y\}$ e si può corallierizzare usando il minimo numero che posso sottrarre a $\min\{x,y\}$ per ottenere un divisore di x,y

$$\text{MCD}(x,y) = \min(x,y) - \mu_{z < \min(x,y)} (1 - \text{div}(\min(x,y)-z, x) \wedge \text{div}(\min(x,y)-z, y))$$

d) numero di divisori primi di x

$$\sum_{z \leq x} \Pr(z) \cdot \text{div}(z, x)$$

* Codifica di coppie (e m-ples)

Vediamo una codifica in \mathbb{N} di coppie (e m-ples) di numeri naturali che risulta poi utile per alcune considerazioni sulla ricorsione (e per il futuro...)

Definiamo come codifica delle coppie

$$\pi : \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$\pi(x, y) = 2^x (2y + 1) - 1$$

Si osservi che π è

- bigettiva
- effettiva (calcolabile!)

L'inversa può essere caratterizzata in termini di due funzioni calcolabili che danno la prima e la seconda componente di un naturale x visto come coppia

$$\pi^{-1} : \mathbb{N} \rightarrow \mathbb{N}^2$$

$$\pi^{-1}(x) = (\pi_1(x), \pi_2(x))$$

dove

$$\pi_1(x) = (x+1)_1$$

$$\pi_2(x) = \left(\frac{x+1}{2^{\pi_1(x)}} - 1 \right) / 2 \quad [\text{la divisione è qt } (-, -)]$$

Si può generalizzare ad una codifica di m-ples

$$\pi^m : \mathbb{N}^m \rightarrow \mathbb{N} \quad m \geq 2$$

definito inducitivamente

$$\pi^2 = \pi$$

$$\pi^{m+1}(\vec{x}, y) = \pi(\pi^m(\vec{x}), y) \quad \vec{x} \in \mathbb{N}^m, y \in \mathbb{N}$$

e corrispondentemente possiamo definire le proiezioni

$$\pi_j^m : \mathbb{N} \rightarrow \mathbb{N}$$

La funzione di Fibonacci è definita da:

$$f(0) = 1$$

$$f(1) = 1$$

$$f(y+2) = f(y) + f(y+1)$$

Questo non è esattamente una definizione per ricorsione primitiva, dato che $f(y+2)$ è definita in termini di $f(y)$ oltre che di $f(y+1)$.

non rispetta lo schema.

Possiamo mostrare che f è calcolabile ricorrendo allo codifico e definendo

$$g: \mathbb{N} \rightarrow \mathbb{N}$$

$$g(y) = \Pi(f(y), f(y+1))$$

quindi g può essere definita per ricorsione primitiva

$$g(0) = \Pi(f(0), f(1)) = \Pi(1, 1)$$

$$\begin{aligned} g(y+1) &= \Pi(f(y+1), f(y+2)) = \Pi(f(y+1), f(y) + f(y+1)) \\ &= \Pi(\Pi_2(g(y)), \Pi_1(g(y)) + \Pi_2(g(y))) \end{aligned}$$

g

g calcolabile

z

$$f(y) = \Pi_1(g(y)) \quad \text{calcolabile}$$

In generale potremmo avere una funzione f definita facendo uso di k valori precedenti

$$\left\{ \begin{array}{l} f(0) = c_0 \\ \vdots \\ f(k-1) = c_{k-1} \end{array} \right.$$

$$\left\{ \begin{array}{l} f(y+k) = h(f(y), \dots, f(y+k-1)) \end{array} \right.$$

$$(f(y+k)) = h(f(y), \dots, f(y+k-1)) \quad \text{con } h: \mathbb{N}^k \rightarrow \mathbb{N} \text{ calcolabile}$$

Si può procedere come prima e definire

$$g: \mathbb{N} \rightarrow \mathbb{N}$$

$$g(y) = \pi^k(f(y), \dots, f(y+k-1))$$

la funzione g può essere definita per ricorsione primitiva

$$g(0) = \pi^k(c_0, \dots, c_{k-1})$$

$$g(y+1) = \pi^k(f(y+1), \dots, f(y+k-1), f(y+k))$$

$$\pi_2^k(g(y)) \quad \pi_k^k(g(y)) \quad h(f(y), \dots, f(y+k-1)) \\ h(\pi_1^k(g(y)), \dots, \pi_k^k(g(y)))$$

$$= \pi^k(\pi_2^k(g(y)), \dots, \pi_k^k(g(y)), h(\pi_1^k(g(y)), \dots, \pi_k^k(g(y))))$$

§

g calcolabile

§

$f(y) = \pi_1(g(y))$ calcolabile

* MINIMALIZZAZIONE

Gli operatori per manipolare funzioni visti finora

- composizione generalizzata
- ricorsione primitiva

a partire da funzioni totali restituiscono funzioni totali.

Un ultimo operatore, in un certo senso (che verrà chiarito) essenziale, che permette di costruire funzioni parziali è l'operat. di mimimalizzazione (illimitata)

Simile alla mimimalizz. limitata, ma... data $f(\bar{x}, y)$ non necessar. totale, definisce $\mu y \cdot f(\bar{x}, y)$ la seguente funzione

$$\mu y \cdot f(\bar{x}, y) = \text{"mimimo } y \text{ tali che } f(\bar{x}, y) = 0"$$

{ma ① se non c'è nessun } y t.c. $f(\bar{x}, y) = 0$ $\rightarrow \uparrow$

e problemi

② se prima di incontrare un y t.c. $f(\bar{x}, y) = 0$
accade che $f(\bar{x}, z) \uparrow$
ma la mimimalizzazione è \uparrow

intuitivo se pensiamo all' ovvio

algoritmo per calcolarlo: im

- inizio da 0: $f(\bar{x}, 0) = 0?$ \rightarrow si out(0)

no $\checkmark f(\bar{x}, 1) = 0?$

finché $f(\bar{x}, y) = 0$

Def. Sia $f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ una funzione. Allora la funzione $h: \mathbb{N}^k \rightarrow \mathbb{N}$ definita per mimimalizzazione (illimitata) è

$$h(\bar{x}) = \mu y \cdot f(\bar{x}, y) = \begin{cases} \text{mimimo } z \text{ t.c. } \begin{cases} f(\bar{x}, z) = 0 & \text{e} \\ f(\bar{x}, z') \downarrow & f(\bar{x}, z') \neq 0 \text{ per } z < z' \end{cases} \\ \text{se un tale } z \text{ esiste} \\ \uparrow \text{ altrimenti} \end{cases}$$

Teorema (Chiusura di \mathcal{C} per minimizzazione)

Sia $f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ una funzione calcolabile (non necess. totale).

Allora $h: \mathbb{N}^k \rightarrow \mathbb{N}$ def. da

$$h(\bar{x}) = \mu y \cdot f(\bar{x}, y)$$

è calcolabile.

dim

Sia F un programma in forma standard per f .

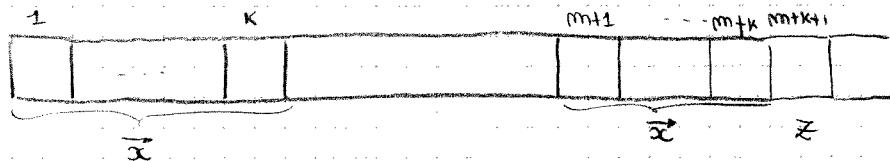
Idea: pur $z = 0, 1, 2, \dots$ si calcola $f(\bar{x}, z)$

finché non si trova uno zero..

Abbiamo bisogno di salvare l'argomento \bar{x} in una zona non "usata" dal programma F .

$$m = \max \{ p(F), k+1 \}$$

quindi il programma per h si ottiene come segue.



$T([1, k], [m+1, m+k+1])$

salva \bar{x}

LOOP: $F [m+1, \dots, m+k+1 \rightarrow 1]$

$f(\bar{x}, z) \rightsquigarrow R_1$

J(1, m+k+2, FINE)

$(m+k+2)$ contiene '0' $\Rightarrow f(\bar{x}, z) = 0$?

S(m+k+1)

$z = z + 1$

J(1, 1, LOOP)

FINE: $T(m+k+1, 1)$

Nota: F può non terminare...

corretto! E' un buon programma
non termina e pu' esplodere!

Nota: While implementato con IP goto.

□

(56)

Oss. L'operatore μ permette di ottenere funzioni non totali a partire da funzioni totali

Esempio: Data $f(x,y) = |x-y^2|$

$$\mu y. f(x,y) = \begin{cases} \sqrt{x} & \text{se } x \text{ quadrato perfetto} \\ \uparrow & \text{altrimenti} \end{cases}$$

Esercizio: Sia $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale e iniettiva.

Allora l'inversa $f^{-1}: \mathbb{N} \rightarrow \mathbb{N}$

$$f^{-1}(x) = \begin{cases} y & \text{se } f(y) = x \\ \uparrow & \text{se } \nexists y \text{ s.t. } f(y) = x \end{cases}$$

è calcolabile

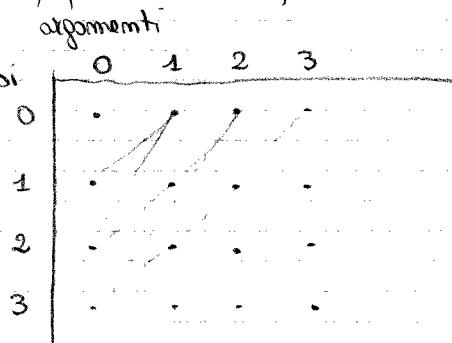
Infatti, sulle nostre ipotesi

$$f^{-1}(x) = \mu y. |f(y)-x|$$

Note: Questa dimostrazione usa in modo essenziale il fatto che f è totale ma il risultato è invece del tutto generale...

Intuitivamente, quando f non totale, per trovare $f^{-1}(x)$

considero un programma P per f passi
e lo eseguo.



code di colomba

0 passi su 0

1 passo su 0

0 passi su 1

2 passi su 0

ogni qualvolta per un certo numero di passi k su argomento y , il programma termina: controllo l'output $f(y)$ ~ se $f(y) = x \rightarrow$ fine
se $f(y) \neq x$ continua.

Informale... vedremo come formalizzarlo

Esercizio: Dimostrare che è computabile la funzione

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

Primo tentativo

$$f(x,y) = \mu z. |y \cdot z - x|$$

va quasi bene ... tranne per il fatto che $f(0,0) = 0$ invece che \uparrow

si può risolvere il problema con un turco (classico)

$$f(x,y) = \mu z. (|y \cdot z - x| + \chi_{x=0 \wedge y=0}(x,y))$$

Esercizio: Tutte le funzioni con dominio finito sono calcolabili

Ovvero:

* Sia $\Theta: \mathbb{N} \rightarrow \mathbb{N}$ $\text{dom}(\Theta)$ finito $\Rightarrow \Theta$ calcolabile

dim

Sia $\Theta: \mathbb{N} \rightarrow \mathbb{N}$ a dominio finito

$$\Theta = \{(x_1, y_1), \dots, (x_m, y_m)\}$$

ovvero

$$\Theta(x) = \begin{cases} y_1 & x = x_1 \\ y_m & x = x_m \\ \uparrow & \text{altrimenti} \end{cases}$$

quindi

$$\Theta(x) = \sum_{i=1}^m y_i \cdot \overline{\text{sg}}(|x - x_i|) + \mu z. \left(\prod_{i=1}^m |x - x_i| \right)$$

serve solo a rendere la funzione \uparrow

quando $x \neq x_1, \dots, x_k$

vale 0 altrimenti.

ALTRI APPROCCI ALLA CALCOLABILITÀ (Tesi di Church)

Abbiamo già osservato che la macchina URM è solo uno dei possibili modelli di calcolo che permettono di formalizzare la nozione di funzione calcolabile.

Avevamo potuto usare:

- Macchina di Turing
- Sistemi di deduzione canonici di Post
- λ-calcolo di Church
- funzioni parziali ricorsive di Gödel-Kleene

Tutti questi approcci definiscono la stessa classe di funzioni calcolabili.

E conduce alla

Tesi di Church: Una funzione è calcolabile tramite un procedimento effettuabile se e solo se è URM-calcolabile.

(versione partigiana)

A comprova della tesi di Church ora vogliamo:

- introdurre un altro formalismo per la definizione di funzione calcolabile
- funzioni parziali ricorsive di Gödel-Kleene → classe R
- dimostrare che è "equivalente" alle URM, ovvero definisce la stessa classe di funzioni.

$$R = C$$

* Funzioni parziali ricorsive

Def: La classe R delle funzioni parziali ricorsive è la più piccola classe di funzioni parziali sui naturali $R \subseteq \bigcup_k \mathbb{N}^k \rightarrow \mathbb{N}$ che contiene

- (a) zero $\Omega: \mathbb{N}^k \rightarrow \mathbb{N}$ $\Omega(\vec{x}) = 0$
- (b) successore $s: \mathbb{N} \rightarrow \mathbb{N}$ $s(x) = x + 1$
- (c) proiezioni $U_i^k: \mathbb{N}^k \rightarrow \mathbb{N}$ $U_i^k(\vec{x}) = x_i$

e chiusa rispetto a

- (1) composizione generalizzata
- (2) ricorrenza primitiva
- (3) minimizzazione illimitata

E' una buona definizione? Cosa significa minima? Esiste una tale classe?

Più precisamente possiamo definire:

Def (Classe ricca): Una classe di funzioni $A \subseteq \bigcup_k \mathbb{N}^k \rightarrow \mathbb{N}$ si dice ricca se

- contiene (a), (b), (c)
- è chiusa rispetto a (1), (2), (3)

Si osserva che la proprietà di essere "classe ricca" è chiusa per intersezione

* Sia $\{A_i\}_{i \in I}$ una famiglia di classi ricche. Allora $\bigcap_{i \in I} A_i$ ricca

e infine si definisce

Def: L'insieme delle funzioni parziali ricordive è

$$R = \bigcap_{\substack{A \subseteq \bigcup_k \mathbb{N}^k \rightarrow \mathbb{N} \\ A \text{ ricca}}} A$$

N.B. R ammette uno scarto induttivo
 $R_0 = \{a, b, c\}$
 $R_m = R_0 \cup \{1, 2, 3 \text{ opp. a } R_i\}$

Altra classe interessante, su cui formuleremo:

$$\rightsquigarrow R = \bigcup_i R_i$$

Def (Funzioni Primitive Ricorsive)

La classe delle funzioni primitive ricorsive è la minima classe di funzioni $PR \subseteq \bigcup_k \mathbb{N}^k \rightarrow \mathbb{N}$ che contiene (a), (b), (c) e chiuse per (1) e (2)

Teorema: $R = C$ { le funz. parziali recursive coincidono con quelle URM-calcolabili }

dimm.

* $R \subseteq C$

Abbiamo dimostrato in precedenza che C è ricco. $\Rightarrow R \subseteq C$

* $C \subseteq R$

sia $f: \mathbb{N}^k \rightarrow \mathbb{N}$ e e una funzione URM-calcolabile

e sia P un programma in forma standard per f

$$P = I_1 \dots I_s$$

Consideriamo le seguenti funzioni (dipendenti da P)

$c_p^1(\vec{x}, t) =$ contenuto di R_1 dopo t passi di computaz. di $P(\vec{x})$
 (ovvero a partire da $\boxed{\vec{x}}$)

un passo di computazione è
 l'esecuzione di una istruz.

Resta inteso che se $P(\vec{x})$ termina in meno di t passi, $c_p^1(\vec{x}, t)$

dà il contenuto di R_1 nella configurazione finale

$j_p(\vec{x}, t) = \begin{cases} \text{numero della istruzione da eseguire al } (t+1)\text{-mo passo} \\ (\text{dopo } t \text{ passi}) \text{ di } P(\vec{x}), \text{ se la comput. non è finita} \\ 0 \quad \text{altrimenti} \end{cases}$

Chiaramente c_p^1 e j_p sono funzioni totali (servirà dopo...)

Se $f(\vec{x}) \downarrow$ allora

$P(\vec{x}) \downarrow$ in un numero di passi pari a $t_0 = \mu t. j_p(\vec{x}, t)$

e

$$f(\vec{x}) = c_p^1(\vec{x}, t_0)$$

altrimenti, se $f(\vec{x}) \uparrow$ anche $P(\vec{x}) \uparrow$ e

$$\mu t. j_p(\vec{x}, t) \uparrow$$

Pectamto

$$f(\vec{x}) = c_p^1 (\vec{x}, \text{pt. } J_p(\vec{x}, t))$$

Dunque per concludere che $f \in \mathbb{R}$ "basta" dimostrare che $\mathbf{c}_p^2, \mathbf{J}_p \in \mathbb{R}$

68

Idea:

- si opera su codifiche di sequenze che rappresentano la configurazione dei registri e program counter
- si manipolano tali sequenze con le funzioni (p_x , q_t , div , ...) che abbiamo costruito per
 - composizione
 - ricorso alme primitiva
- così si ottengono c_p^1, j_p dalle suddette funzioni, per ricorso alme primitiva

Più precisamente: lo stato della computaz. è una coppia $\langle \vec{R}, t \rangle$

* una configurazione dei registri

$$z_1 | z_2 | \dots$$

nella quale al più un numero finito di registri contiene un valore $\neq 0$

۷

si può codificare con

$$\text{cod}(\vec{R}) = \prod_{i \geq 1} p_i^{e_i}$$

\nwarrow moto che solo un numero finito di fattori è $\neq 1$

il prodotto potrebbe andare solo fino a $\max\{i \mid z_i \neq 0\}$

a questo punto, dato $x \geq 1$ interpretalo come $\text{cod}(\vec{R})$

$$x_i = (x)_i$$

Utilizzando questa codifica, possiamo considerare la funzione

$c_p(\vec{x}, t)$ = configurazione dei registri dopo t passi di $P(\vec{x})$

come una funzione

$$c_p : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$$

Si può dunque definire

$$\sigma_p(\vec{x}, t) = \langle c_p(\vec{x}, t), j_p(\vec{x}, t) \rangle$$

= "stato della computazione $P(\vec{x})$ dopo t passi"

e utilizzando la funzione di codifica per le copie π possiamo vedere σ_p come

$$\sigma_p : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$$

$$\sigma_p(\vec{x}, t) = \pi(c_p(\vec{x}, t), j_p(\vec{x}, t))$$

e possiamo definirla per ricorsione primitiva

$$\sigma_p(\vec{x}, 0) = \pi(\prod_{i=1}^k p_i^{x_i}, 1)$$

$$\sigma_p(\vec{x}, t+1) = \pi(c_p(\vec{x}, t+1), j_p(\vec{x}, t+1))$$

con

$$q_t(p_m^{(c_p(\vec{x}, t))_m}, c_p(\vec{x}, t)) \quad \text{se } 1 \leq j_p(\vec{x}, t) \leq s \\ \circ I_{j_p(\vec{x}, t)} = z(m)$$

$$p_m \cdot c_p(\vec{x}, t) \quad \text{se } 1 \leq j_p(\vec{x}, t) \leq s \\ \circ I_{j_p(\vec{x}, t)} = s(m)$$

$$c_p(\vec{x}, t+1) = q_t(p_m^{(c_p(\vec{x}, t))_m}, c_p(\vec{x}, t)) \cdot p_m^{(c_p(\vec{x}, t))_m} \quad \text{se } 1 \leq j_p(\vec{x}, t) \leq s \\ \circ I_{j_p(\vec{x}, t)} = T(m, m)$$

$$c_p(\vec{x}, t) \quad \text{altrimenti, ovvero}$$

$$1 \leq j_p(\vec{x}, t) \leq s \wedge I_{j_p(\vec{x}, t)} = J(m, m, 0)$$

$$\text{oppure } J_p(m, m, t) = 0$$

$$J_p(\vec{x}, t+1) = \begin{cases} J_p(\vec{x}, t) + 1 & \text{se } 1 \leq J_p(\vec{x}, t) < S \text{ e} \\ & I_{J_p(\vec{x}, t)} \in Z(m), S(m), T(m, m) \text{ o} \\ & J(m, m, q) \text{ com.} \\ & (\epsilon_p(\vec{x}, t))_m \neq (\epsilon_p(\vec{x}, t))_m \\ q & \text{se } 1 \leq J_p(\vec{x}, t) \leq S \text{ e} \\ & I_{J_p(\vec{x}, t)} = J(m, m, q) \text{ com. } q \leq S \text{ e} \\ & (\epsilon_p(\vec{x}, t))_m = (\epsilon_p(\vec{x}, t))_m \\ 0 & \text{altrimenti} \end{cases}$$

anche se non formalizzato, un pieno ottoglio mi definizione per ricorsione primitiva di ϵ_p a parziale da funzioni in PR

$$\Rightarrow \epsilon_p \in \text{PR} \quad \{ \text{NOTA: tutte le funzioni usate sono in PR} \Rightarrow \epsilon_p \in \text{PR} \}$$

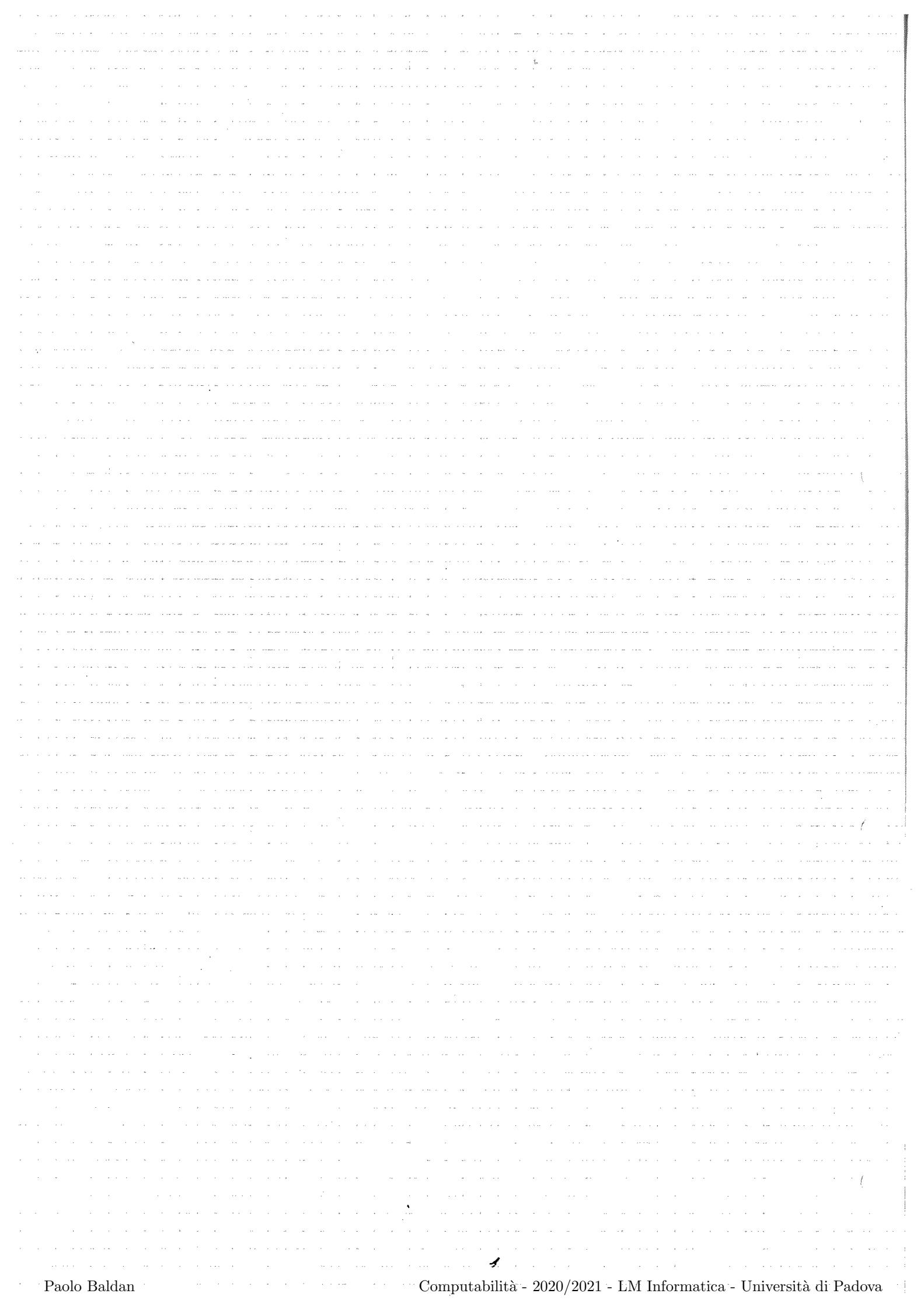
es

$$c_p^1(\vec{x}, t) = (\Pi_1(\epsilon_p(\vec{x}, t)))_1 \in \text{R} \quad (\in \text{PR})$$

$$J_p(\vec{x}, t) = \Pi_2(\epsilon_p(\vec{x}, t))$$

e dunque f definita \times composizione / minimizz. de c_p^1, J_p
 è in R , come desiderato

□



FUNZIONI PRIMITIVE RICORSIVE

Torniamo a considerare la classe delle funzioni primitive ricorsive.

Def. La classe delle funzioni primitive ricorsive è la più piccola classe

$$\text{PR} \subseteq \bigcup_{k \in \mathbb{N}} (\mathbb{N}^k \rightarrow \mathbb{N})$$

comtemente

- (a) zero
- (b) successore
- (c) proiezioni

e chiusa per

- (d) composizione
- (e) ricorsione primitiva

Ci sono vari motivi di interesse per PR ... ad esempio il fatto che

- ricorsione primitiva \rightsquigarrow for
- minimizzazione \rightsquigarrow while

Questo si può formalizzare considerando una variante della macchina URM con programmi strutturati, dove l'istruzione di salto è sostituita dai costrutti for, while ... URM_{for,while}

(questo corrisponde a considerare solo un sottoinsieme dei programmi URM in cui i $j(-)$ sono usati in modo "disciplinato".)

Si dimostra che il modello ha lo stesso espressività della URM, ovvero se $E_{\text{for,while}}$ è la classe delle funzioni calcolabili nel modello

$$E_{\text{for,while}} = E = \mathbb{R}$$

mentre le funzioni calcolabili solo con il for sono le funzioni parziali ricorsive

$$E_{\text{for}} = \text{PR}$$

Quindi studiare la relazione tra \mathbb{R} e PR corrisponde a studiare la relazione tra il potere espressivo dei costrutti for e while.

Sappiamo che sono im PR molte funzioni "aritmetiche"

$P(x)$, $(x)_y$, div, qt, $\text{mcm}(x,y)$; x^y

e che PR è chiusa per

somma		}
prodotto		

minimizzazione

limitati

3

è una classe molto estesa, ma [chiedi] non comincia tutte le funzioni calcolabili, ovvero

$$PR \subseteq R$$

inclusione stretta

Perché? Le funzioni im PR sono tutte totali!

(secondo una coratl. induktiva le funzioni im PR sono ottimibili da funzioni di base totali per composizione e ric.primitiva)

Si potrebbe pensare che che PR comprenda tutte le funzioni ricursivei totali ovvero, se Tot è la classe delle funzioni totali

$$PR = R \cap \text{Tot} \quad [\text{Hilbert 1926}]$$

Anche questo è falso ($PR \subsetneq R \cap \text{Tot}$) ovvero anche se ci restringiamo a funzioni totali (proprio anni che terminiamo sempre) il while è essenziale

5

Funzione di Ackermann (Wilhelm Friedrich Ackermann - 1928)

Una funzione $\psi : \mathbb{N}^2 \rightarrow \mathbb{N}$ definita da

$$\psi(0, y) = y + 1$$

$$\psi(x+1, 0) = \psi(x, 1)$$

$$\psi(x+1, y+1) = \psi(x, \psi(x+1, y))$$

Definisce davvero una funzione? Sì!

Formalmente ψ è il minimo punto fisso dell'operatore associato alla definizione ricorsiva, sul cpo delle funzioni parziali

$$T : (\mathbb{N}^2 \rightarrow \mathbb{N}) \rightarrow (\mathbb{N}^2 \rightarrow \mathbb{N})$$

Più intuitivamente, lo schema individua univocamente una funzione perché il valore di $\psi(x, y)$ è sempre definito im termini del valore di ψ stessa su argomenti "più piccoli".

In che senso "più piccoli"?

$$\psi(x+1, 0) = \psi(x, 1) \quad \text{vi diminuisce la prima componente}$$

$$\psi(x+1, y+1) = \sim \text{prima colonna } \psi(x+1, y), \text{ diminuisce la 2a} \\ \text{componente} \\ \text{e quindi} \quad \psi(x, u)$$

qualunque sia u , la prima componente è diminuita

gli argomenti diminuiscono nell'ordine lessicografico su \mathbb{N}^2

$$(\mathbb{N}^2, \leq_{lex}) \quad (x, y) \leq_{lex} (x', y') \quad \text{se } (x < x') \quad \vee \\ (x = x' \text{ e } y \leq y')$$

bene ordinato \leadsto non contiene catene discendenti infinite!

Parentesi su ordini parziali

Def. Sia (D, \leq) un ordine parziale.

- bene ordinato se ogni sottoinsieme non vuoto ha minimo
 $\forall X \subseteq D, X \neq \emptyset \Rightarrow \exists \text{min } D$
- bem fondato se ogni sottoinsieme non vuoto ha un elemento minimaile
 $\forall X \subseteq D, X \neq \emptyset \Rightarrow \exists \text{dex minimaile}$

OSSERVAZIONE: D bene ordinato $\Rightarrow D$ ben fondato

OSSERVAZIONE: se D è ben fondato non contiene catene discendenti infinite
 $d_0 > d_1 > d_2 > \dots > d_m > d_{m+1} > \dots$

Questo fatto è spesso utile nella soluzione di problemi di terminazione.
se si riesce ad attribuire all'insieme delle configurazioni una struttura d'ordine ben fondato, per concludere basta provare che ad ogni posso

comfi \rightarrow comfi_{i+1}

vale

comfi_{i+1} < comfi

68 e che quindi la computazione per colui uno stesso discendente, memoriamente finito.

Nello specifico, il calcolo di ψ è ricordato sempre al calcolo di ψ stessa su valori più piccoli e pensato come un procedimento, fermi ma!

si noti che $(\mathbb{N}^2, \leq_{lex})$ è bene ordinato

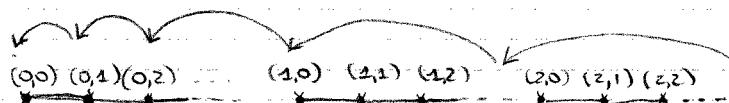
(dato $\phi \neq X \subseteq \mathbb{N}^2$, detto $x_0 = \min \{x \mid \exists y. (x,y) \in X\}$
 $y_0 = \min \{y \mid (x_0, y) \in X\}$)

allora

$$\min X = (x_0, y_0)$$

così si dimostra, più in generale che il prodotto di due poset ben ordinati è ben ordinato)

ma l'ordine è totale (esercizio)



- "infinte copie di \mathbb{N} una di seguito all'altra..."

- ogni elemento ha un successore ma non sempre un predecessore

- percorrendo l'ordine all'indietro si formano dei solchi di lunghezza infinita.

Associata ad un ordine ben fondato vi è un principio di induzione

* Induzione ben fondata

Sia (D, \leq) ordine ben fondato e sia $P \subseteq D$ una proprietà

se $\forall d \in D . (\forall d' < d . P(d')) \Rightarrow P(d)$



(*)

$\forall d \in D . P(d)$

ovvero per dimostrare che P vale su tutto D basta dimostrare che

- vale sugli elementi minimali

- per ogni altro elemento d , se vale sui più piccoli di d vale anche su d

Esercizio. fare la dimostrazione.

Terminiamo alla funzione di Ackermann ψ che è molto interamente perché è

- 1) totale
- 2) calcolabile $(\rightsquigarrow \text{PR} \subseteq \text{RinTot})$
- 3) non primitiva ricorsiva

1) ψ è totale

Si può dimostrare per induzione binaria fondata su $(\mathbb{N}^2, \leq_{\text{lex}})$ che

$$\forall (x,y) \in \mathbb{N}^2 \quad \psi(x,y) \downarrow$$

dim

- elementi minimali

In realtà $(\mathbb{N}^2, \leq_{\text{lex}})$ ha un unico minimo $(0,0)$ e

$$\psi(0,0) = 1 \downarrow$$

- passo induttivo: $\forall (x,y) \in \mathbb{N}^2 . ((\forall (x',y') \in \mathbb{N}^2 . ((x',y') <_{\text{lex}} (x,y) \rightarrow \psi(x',y') \downarrow) \Rightarrow \psi(x,y) \downarrow)$

$$(x=0) \quad \psi(0,y) = y+1 \downarrow$$

$(x>0)$ due sottocasi

$$(y=0) \quad \psi(x,0) = \psi(x-1, 1) \downarrow$$

\times ip. ind. poiché $(x-1, 1) <_{\text{lex}} (x, 0)$

$$(y>0) \quad \psi(x,y) = \psi(x-1, \psi(x,y-1))$$

$$\psi(x,y-1) \downarrow \quad \times \text{ ip. ind.}$$

$$\text{e dunque } u = \psi(x,y-1)$$

$$\psi(x,y) = \psi(x-1, u) \downarrow \quad \times \text{ ip. ind.}$$

Esercizio: Data una urna che contiene un numero arbitrario di palline, etichettate con un numero in \mathbb{N} , si opera come segue:
 - si estrae una pallina, la si butta e sostituisce con un numero arbitrario di palline con etichetta < (se 0, la si butta e basta!).
 Dimostrare che il processo termina sempre.

2) ψ è calcolabile ($\psi \in \mathbb{R} = \mathbb{C}$)

Potremmo appellarsi alla tesi di Church: il calcolo di $\psi(x,y)$ è sempre ridotto al calcolo di ψ sui valori più piccoli e nel caso base uso il successore.

Si può formalizzare come segue

- * insieme $S \subseteq \mathbb{N}^3$ valido intuitivamente se
 - $(x,y,z) \in S \Rightarrow z = \psi(x,y)$
 - $(x,y,z) \in S \Rightarrow S$ contiene tutte le triple $(x',y',\psi(x',y'))$ necessarie per il calcolo di $\psi(x,y)$

$$\text{es: } \psi(1,1) = \underbrace{\psi(0, \underbrace{\psi(1,0)}_{\psi(0,1)})}_{\psi(0,2)} = \underbrace{\psi(0, \underbrace{\psi(0,1)}_2)}_{\psi(0,2)} = \psi(0,2) = 3$$

$$\rightsquigarrow (1,1,3) \quad (0,2,3) \quad (1,0,2) \quad (0,1,2)$$

Def Un insieme di triple $S \subseteq \mathbb{N}^3$ si dice valido se

- $(0,y,z) \in S \Rightarrow z = y+1$
- $(x+1,0,z) \in S \Rightarrow (x,1,z) \in S$
- $(x+1,y+1,z) \in S \Rightarrow \exists u. (x+1,y,u) \in S \wedge (x,u,z) \in S$

Non è difficile dimostrare che:

- * $\psi(x,y) = z \Leftrightarrow \exists S \subseteq \mathbb{N}^3$ valido e finito t.c. $(x,y,z) \in S$

(induzione ben fondata su (x,y) , usando il fatto che la validità è preservata per unione - Esercizio)

Ora, una tripla (x,y,z) può essere codificata con un intero mediante le funzioni di codifica

$$\Pi^3 : \mathbb{N}^3 \rightarrow \mathbb{N} \quad (\Pi_i^3 : \mathbb{N} \rightarrow \mathbb{N} \text{ proiezioni})$$

↑ omettiamo il 3!

quindi un insieme di triple diventa un insieme di naturali $\{(x_1, \dots, x_m)\}$ che si può codificare intuitivamente

$$\{(x_1, \dots, x_m)\} \mapsto p_{x_1} \cdots p_{x_m}$$

Ora, detto σ un numero che codifica l'insieme di triple S_N vale

$$(x, y, z) \in S_N \quad \text{sse} \quad P_{\pi(x, y, z)} \mid \sigma$$

e il predicato

$\text{Val}(\sigma) = " \sigma \text{ codifica un insieme valido di triple"$
è decidibile.

Infatti vale $\text{Val}(\sigma)$ se

$$- \forall i \leq N \quad (\sigma)_i = 1$$

$$- \forall \omega \leq N \quad P_{\omega \mid \sigma} \Rightarrow$$

$$\Rightarrow \begin{cases} \cdot \pi_1(\omega) = 0 \Rightarrow \pi_3(\omega) = \pi_2(\omega) + 1 \\ \cdot \pi_1(\omega) > 0 \end{cases}$$

$$\begin{cases} \cdot \pi_2(\omega) = 0 \Rightarrow \pi(\pi_1(\omega), 0, \pi_3(\omega)) \in S_\omega \\ \cdot \pi_2(\omega) > 0 \Rightarrow \exists u \leq \omega \text{ t.c.} \end{cases}$$

$$\pi(\pi_1(\omega), \pi_2(\omega)-1, u) \in S_\omega$$

$$\text{e } \pi(\pi_1(\omega)-1, u, \underline{z}) \in S_\omega$$

e la corrispondente funz. caratteristica.

$$\chi_{\text{val}} \in \text{PR}$$

Ora, possiamo verificare che è calcolabile ($\in \text{PR}$)

$$R(x, y, \omega) = \begin{cases} \text{se } \omega \text{ è lo codifica di un } S \text{ valido} \\ \text{che contiene } (x, y, z) \text{ per qualche } z \\ \text{o altrimenti} \end{cases}$$

$$R(x, y, \omega) = \chi_{\text{val}(\omega)} \cdot \text{sg}(\omega + 1 - \mu z \leq \omega \cdot \text{div}(P_{\pi(x, y, z)}, \omega))$$

↳

possiamo esprimere la funzione di Ackermann come

$$\psi(x, y) = \mu(z, w) \cdot \text{sg}(R(x, y, \omega) \cdot \text{div}(P_{\pi(x, y, z)}, \omega))$$

uso μ illimitato!

codifica

3) $\Psi \notin PR$

Il fatto che la definizione di Ψ include una ricorsione omnidata (e quindi non rispetta lo schema di ricorsione primitiva) non ci permette di concludere...

La dimostrazione del fatto che Ψ non è primitiva ricorsiva, intuitivamente, mostra che Ψ cresce più velocemente di qualsiasi funzione in PR.

Abbiamo visto come:

- somma si ottiene dal successore
- prodotto " " " somma
- esponenziale " " " prodotto

} per ricorsione primitiva

amidata sempre un numero maggiore di volte

L'idea della funzione di Ackermann è quella di portare al limite questo procedimento e costruire ogni possibile livello in una unica funzione che quindi non potrà essere espressa con nessun ordinamento finito di ric. primitive.

Infatti, indicato con

$$\Psi_x(y) = \Psi(x, y)$$

si ha

$$\Psi_x(\Psi_{x+1}(y-2))$$

$$\begin{aligned} \Psi_{x+1}(y) &= \Psi_x(\overbrace{\Psi_{x+1}(y-1)}^{\Psi_x(\Psi_{x+1}(y-2))}) = \Psi_x^2(\Psi_{x+1}(y-2)) = \\ &= \underbrace{\Psi_x^y(\Psi_{x+1}(0))}_{\Psi_x(1)} = \Psi_x^{y+1}(1) \end{aligned}$$

$$\Psi_0(y) = y+1 = \text{succ}(y)$$

$$\Psi_1(y) = \Psi_0^{y+1}(1) = \text{succ}^{y+1}(1) = y+2$$

$$\Psi_2(y) = \Psi_1^{y+1}(1) = 2y+3$$

$$\Psi_3(y) = \Psi_2^{y+1}(1) = 2^{(y+3)} - 3$$

$$\Psi_4(y) = \Psi_3^{y+1}(1) = 2^{2^{2^{y+3}}} - 3$$

es:

$$\psi_0(1) = 2$$

$$\psi_1(1) = 3$$

$$\psi_2(1) = 5$$

$$\psi_3(1) = 13$$

$$\psi_4(1) = 2^{2^2} - 3 = 2^{16} - 3 \quad \curvearrowright 64$$

$$\psi_4(2) = 2^{2^4} - 3 \quad \approx 10^{3200}$$

$$\psi_4(3) = 2^{2^6} - 3 \quad \approx 2^{10^{3200}}$$

Intuitivamente, al crescere di x in $\psi_x(y)$ cresce il livello di ammidenamento \Rightarrow # di fasi omnidati necessari per il calcolo

↳

dato che x può crescere illimitatamente non si può calcolare ψ utilizzando solo dei for \Rightarrow occorre un while!

$\Rightarrow \psi \notin PR$

Più precisamente si può mostrare che per ogni funzione $f \in PR \Rightarrow$ calcolata da un programma P che usa solo for detto j il massimo ammidenamento di cicli for, vale assolutamente

$$P(x_1, \dots, x_k) \downarrow \text{in un numero di passi} \leq \psi_{j+1}(\max\{x_1, x_k\})$$

↳

ψ_j dà un bound alla complessità in tempo

questo è anche un limite superiore al # di operazioni di incremento quindi

$$f(\vec{x}) \leq \psi_{j+1}(\max(\vec{x}))$$

Pertanto, se $\psi \in PR$, detto j il massimo ammidenamento di cicli for in un programma che lo calcola, dovrebbe valere

$$\psi(x, y) \leq \psi_{j+1}(\max\{x, y\})$$

I'm pericolare con $x = y = k > j + 1$, abbastanza grande

$$\psi(k, k) < \psi_{j+1}(k) < \varphi_k(k) = \varphi(k, k)$$

osserva.

omwdo!

OSSERVAZIONE: Gödel e Kleene inizialmente avevano studiato una classe di funzioni Ro , dette μ -ricorsive

complementi

(a) zero

(b) successore

(c) proiezioni

e chiuse rispetto a

(1) composizione

(2) ricorsione primitiva

(3) minimizzazione ristretta al caso in cui la funzione che produce è totale.

E' chiaro che

$$\text{Ro} \subseteq \text{R}$$

e l'inclusione è stretta dato che

- le funzioni in Ro sono tutte totali

- alcune funzioni in R sono parziali

Amoore

$$\text{Ro} \subseteq \text{R} \cap \text{Tot}$$

ma non è ovvio che valga l'uguaglianza!

Infatti, una funzione in $\text{R} \cap \text{Tot}$ può essere totale, ma ottenuta per minimizzazione di funzioni parziali

Es:

$$f(x, y) = \begin{cases} x+1 & x < y \\ 0 & x = y \\ 1 & x > y \end{cases}$$

$$\exists y \ f(xy) = 1 \forall x$$

in generale non è vero che se $f(x,y)$ è parziale e $\mu_y f(x,y)$

totale allora

$$\mu_y f(x,y) \in R_o$$

ma vale!

Teorema : $R_o = R \cap \text{Tot}$

dim

(\subseteq) ovvio

(\supseteq) sia $f \in R \cap \text{Tot}$ in $f \in e$ \Rightarrow (x dim. di $R = e$)

$$f(\bar{x}) = \wp^1(\bar{x}, \mu_t. j_p(\bar{x}, t))$$

f_R \leftarrow $R \rightarrow \text{totale}$
totale

□

ENUMERAZIONE DELLE FUNZIONI CALCOLABILI

L'obiettivo è quello di definire una enumerazione "effettiva" dei programmi URM e delle funzioni (URM-) calcolabili

3 sarà fondamentale per la nostra teoria ed in particolare per

- dim. esistenza di funzioni non calcolabili (\sim alla dim. informale già vista, ma qui le sapremo anche "costruire")

- teorema S-m-m (o del parametro)

- funzione / macchina universale

- Ripasso su cardinalità

dati due insiemi A, B

- $|A|=|B|$ ne esiste una funzione $f: A \rightarrow B$ bigettiva.

(corrispondenza biunivoca)

- $|A| \leq |B|$ se esiste una funzione $f: A \rightarrow B$ iniettiva.

oppure $g: B \rightarrow A$ surgettiva

- se $|A| \leq |B|$ e $|B| \leq |A| \Rightarrow |A|=|B|$

(enunciato il principio della scelta)

$\{A_i\}_{i \in I}$ famiglia di insiemi non vuoti $\forall i \in I \quad A_i \neq \emptyset$

allora esiste una funzione

$$f: I \rightarrow \bigcup_{i \in I} A_i \quad (\text{c. } \forall i \in I \quad f(i) \in A_i)$$

* Un insieme X si dice numerabile se $|X| \leq |\mathbb{N}|$, ovvero esiste

$f: \mathbb{N} \rightarrow X$ surgettiva

In questo caso f si dice una enumerazione di X , intuitivamente perché posso estrarre tutti gli elem di X come

$f(0) \quad f(1) \quad f(2) \quad \dots$

ci sono tutti!

\Rightarrow se f è anche iniettiva (\rightarrow biunivoco) si dice una enumerazione senza ripetizioni

enumerazione effettiva di X : enumerazione $f: \mathbb{N} \rightarrow X$ tale che

f e f^{-1} sono effettive

(quando $X = \mathbb{N}^k$)

$f^{-1}: X \rightarrow \mathbb{N}$ può essere mostrata calcolabile

altrimenti ricorriamo alla nozione intuitiva di effettività)

Quando l'enumerazione è bimivoca spesso la orientiamo in senso inverso

$$X \rightarrow \mathbb{N}$$

Un po' di risultati preliminari

Lemma: I seguenti insiemi sono effettivamente enumerabili

(con corrispondenze bimivoca effettive)

$$(1) \quad \mathbb{N}^2$$

$$(2) \quad \mathbb{N}^3$$

$$(3) \quad \bigcup_{K \geq 2} \mathbb{N}^K$$

dim

(1) Abbiamo già visto la biezione

$$\pi: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$\pi(x, y) = 2^x(2y+1)-1$$

calcolabile ($\in PR$) con inversa

$$\pi^{-1}: \mathbb{N} \rightarrow \mathbb{N}^2$$

$$\pi(x) = (\pi_1(x), \pi_2(x))$$

$\pi_1, \pi_2 \in PR$ (\rightarrow calcolabili)

(b) Possiamo considerare

$$\nu: \mathbb{N}^3 \rightarrow \mathbb{N}$$

$$\nu(x, y, z) = \pi(\pi(x, y), z) \quad \forall x, y, z \in \mathbb{N}$$

com inversa realizzata mediante proiezioni

$$\nu^{-1}: \mathbb{N} \rightarrow \mathbb{N}^3$$

$$\nu^{-1}(x) = (\nu_1(x), \nu_2(x), \nu_3(x))$$

$$\nu_1(x) = \pi_1(\pi_1(x))$$

$$\nu_2(x) = \pi_1(\pi_2(x))$$

$$\nu_3(x) = \pi_2(x)$$

$v_1, v_2, v_3 \in PR \rightarrow$ calcolabili $\sim v^{-1}$ effettiva

(c) Si osservi che la codifica

$$(a_1, \dots, a_k) \mapsto \prod_{i=1}^k p_i^{a_i+1}$$

non va bene: imiettiva, ma non surgettiva!

(non sono nell'immagine 0, 1 e nemmeno numero $x = t_c$)

$$\exists i, j \quad i < j \quad p_i | x \text{ e } p_j | x$$

L'idea è quella di sfruttare l'umicità della rappresentazione dei numeri naturali in forma binaria (posizionale)

$$\tau: \bigcup_{k \geq 1} \mathbb{N}^k \rightarrow \mathbb{N}$$

$$\tau(a_1, \dots, a_k) = 2^{a_1} + 2^{a_1+a_2+1} + \dots + 2^{a_1+\dots+a_k+(k-1)} - 1$$

$$= \sum_{i=1}^k \frac{2 \left(\sum_{j=1}^i a_j \right) + (i-1)}{2} - 1 \quad (*)$$

• τ è (intuitivamente) effettiva (ma solo funzione calcolabili), ma il dominio non permette di dim. formalmente la calcolabilità

• τ è una biezione (come chiesto dalla discussione all'inverso che segue)

sia $x \in \mathbb{N}$ vogliamo determinare (a_1, \dots, a_k) t.c. $\tau(a_1, \dots, a_k) = x$

per l'umicità della rappresentazione in forma binaria (di $a+1$)

$$x = (d_m d_{m-1} \dots d_1 d_0)_2 - 1 = d_m 2^m + \dots + d_1 2^1 + d_0 - 1$$

$$d_i \in \{0, 1\}$$

possiamo facilmente definire una funzione $d(x, j)$ t.c.

$$d(x, j) = d_j$$

infatti

$$d(x, j) = \text{rem}(2^j, q_t(2^j, x+1))$$

$$\in PR \subseteq e$$

ora, dalla definizione di τ

$$\tau(x, \neg x) = (1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \dots -1 \ 0 \ 0 \ 1 \ 0 \ 0)$$

$\uparrow \quad \uparrow$
 $a_1 + a_2 + \dots + a_k + k - 1$
 b_k

$\uparrow \quad \uparrow$
 $a_1 + a_2 + 1 \quad a_1$
 $b_2 \quad b_1$

quindi già possiamo esprimere la lunghezza k della tupla come

$$l(x) = \sum_{j \leq x} d(x, j)$$

il numero delle cifre = 1

noto che $x+1$ ha al massimo $x+1$ cifre binarie

Ora considerando solo le cifre a 1 e indicando con b_i le loro posizioni
 (vedi sopra) non è difficile definire una funzione $b(x, i)$ tc

$$b(x, i) = b_i$$

infatti

$$b(x, 1) = \mu j \leq x. "d(x, j) = 1"$$

$$= \mu j \leq x. |d(x, j) - 1|$$

$$b(x, i+1) = \mu j \leq x. "d(x, j) = 1" \text{ e } "j > b(x, i)"$$

$$= \mu j \leq x. (|d(x, j) - 1| + (b(x, i) + 1 - j))$$

EPR!

Infine, possiamo scrivere una funzione $a(x, i)$ tc $a(x, i) = a_i$

$$a(x, 1) = b(x, 1)$$

EPR

$$a(x, i+1) = (b(x, i+1) = b(x, i)) = 1$$

?

l'inversa può finalmente essere espressa, per $x \in N$ come

$$\tau^{-1}(x) = (a(x, 1), \dots, a(x, l(x)))$$

ed è effettiva!

Codifica alternativa delle tuple

$$\tau: \bigcup_{K \geq 1} \mathbb{N}^K \rightarrow \mathbb{N}$$

$$\tau(a_1, \dots, a_K) = \prod_{i=1}^K p_i^{a_i}$$

non funziona, perché identifica $(1, 4)$
 $(1, 4, 0, 0)$

"dimentica" gli zeri finali

Idea: "imamente" l'ultima componente

$$\tau(a_1, \dots, a_K) = \prod_{i=1}^{K-1} p_i^{a_i} \cdot p_K^{a_{K+1}} - 2$$

Inversa: dato $x \in \mathbb{N}$ da intendersi come codifica di una K -pla

* lunghezza: $\ell(x) = \max K \cdot p_K \mid (x+2)$

$$\{ \mu K \leq x, p_{K+1} \nmid x+2 \} \text{ no!!}$$

$$= x - \mu h \leq x, p_{x-h} \mid (x+2)$$

* componenti $a(x, i) = \begin{cases} (x+2)_i & i = 1, \dots, \ell(x)-1 \\ \cdot (x+2)_i - 1 & i = \ell(x) \end{cases}$

* Alternativa: $\tau(a_1, \dots, a_K) = \prod \left(\prod_{i=1}^K p_i^{a_i}, K \right)$
 $\ell(x) = \pi_2(x)$
 $a(x, i) = (\pi_1(x))_i$

Notazione: J indichiamo con

J l'insieme delle istruzioni URM

P " " dei programmi URM

Teorema: P è effettivamente enumerabile

(con una corrispondenza biunivoca effettiva)

dim

1) dimostriamo che esiste una corrispondenza biunivoca effettiva

$$\beta: \mathcal{Y} \rightarrow \mathbb{N}$$

idea: uso le enumerazioni di coppie e triple, e mando

- istruzioni $Z \rightsquigarrow$ multipli di 4

- " " $S \rightsquigarrow$ numeri $\equiv 1$ modulo 4

- " " $T \rightsquigarrow$ " $\equiv 2$ " 4

- " " $J \rightsquigarrow$ " $\equiv 3$ " 4

Z

$$\beta(Z(m)) = 4 * (m-1)$$

$$\beta(S(m)) = 4 * (m-1) + 1$$

$$\beta(T(m, m)) = 4 * \pi_1(m-1, m-1) + 2$$

$$\beta(J(m, m, t)) = 4 * \pi_2(m-1, m-1, t-1) + 3$$

com. inversa

$$\beta^{-1}: \mathbb{N} \rightarrow \mathcal{Y}$$

difinito come segue: dato $x \in \mathbb{N}$ nia $t = \text{rem}(4, x)$

$$q = qt(4, x)$$

allora

$$\beta^{-1}(x) = \begin{cases} Z(q+1) & \text{se } t=0 \\ S(q+1) & \text{se } t=1 \\ T(\pi_1(q)+1, \pi_2(q)+1) & \text{se } t=2 \\ J(V_1(q)+1, V_2(q)+1, V_3(q)+1) & \text{se } t=3 \end{cases}$$

chiaramente β e β^{-1} sono effettive

2) A questo punto, per definire una corrispondenza biunivoca effettiva $\gamma: P \rightarrow \mathbb{N}$ è sufficiente osservare che un programma è una sequenza finita di

Istruzioni, che tramite β può essere trasformata in una sequenza finita di numeri naturali

$$\gamma: \mathbb{P} \rightarrow \mathbb{N}$$

dato $P \in \mathbb{P}$ $P = I_1 \dots I_s$

$$\gamma(P) = \tau(\beta(I_1), \dots, \beta(I_s))$$

5

γ è biunivoca (composizione di funzioni biunivoci)

e γ, γ^{-1} effettive (intuitivamente) in quanto composizioni di funzioni effettive

6

\mathbb{P} è effettivamente enumerabile.

□

Def Dato $P \in \mathbb{P}$ il valore $\gamma(P)$ è detto codice (Gödel number) di P

e spesso scriviamo P_m per indicare $\gamma^{-1}(m)$ (m^{mo} programma nella numerazione)

Note: D'ora in poi faremo sempre riferimento, spesso implicitamente, alla enumerazione di programmi γ che si intende fissata, e quindi determinata il significato di P_m .

L'enumerazione fissata poteva essere definita in modo diverso (basta fissarne una) ma i fondamentali che γ e γ^{-1} siano effettive ovvero che

- dato un programma P sia possibile trovare in modo effettivo il suo codice $\gamma(P)$

- dato un numero m sia possibile determinare effettivamente l' m -mo programma $P_m = \gamma^{-1}(m)$

e si noti che (anche se con un pizzico di informalità) non vi sono dubbi sulla effettività di γ e γ^{-1}

Esempio:

Si consideri il programma P

$T(1,2)$

$S(2)$

$T(2,1)$

è codificato da

$$\beta(T(1,2)) = 4 * \pi(1-1, 2-1) + 2 = 4 * \pi(0,1) + 2 = 10$$

$$\beta(S(2)) = 4 * (2-1) + 1 = 5$$

$$\beta(T(2,1)) = 4 * \pi(2-1, 1-1) + 2 = 4 * \pi(1,0) + 2 = 6$$

3

$$x(P) = \tau(10, 5, 6) = 2^{10} + 2^{\underline{10+5+2}} + 2^{\underline{10+5+6+2}} - 1$$

$$= 2^{10} + 2^{16} + 2^{23} - 1$$

$$= 8455167$$

$$\left[\text{codifica alternativa: } p_2^{10} \cdot p_2^5 \cdot p_3^{6+1} - 2 = 2^{10} \cdot 3^5 \cdot 5^7 - 2 \right]$$

$$= 19439999998$$

Cosa calcola? $\lambda x x+1$

Anche P'

$S(1)$

$$\text{In questo caso } \beta(S(1)) = 4 * (1-1) + 1 = 1$$

$$\sim x(P') = \tau(1) = 2^1 - 1 = 1$$

$$\left[\text{codifica alternativa: } 2^{1+1} - 2 = 2^2 - 2 = 2 \right]$$

Esempio: Determinare P_{100} ovvero $x^{-1}(100)$

Si osserva

$$100 = \begin{array}{ccccccccc} & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 100 = & 1 & 1 & 0 & 0 & 1 & 0 & 1 & - 1 \\ & | & | & | & | & | & | & \\ & b_4 & b_3 & b_2 & b_1 & & & \end{array}$$

$\rightarrow 100$ è la codifica di una quadrupla (programma con 4 istruz)

$$\tau^{-1}(100) = (a_1 a_2 a_3 a_4)$$

Le varie componenti possono essere determinate come visto in prec.

$$a_1 = b_1 = 0$$

$$\rightsquigarrow Z(1)$$

$$a_2 = b_2 - b_1 - 1 = 1$$

$$\rightsquigarrow S(1)$$

$$a_3 = b_3 - b_2 - 1 = 2$$

$$\rightsquigarrow T(1,1)$$

$$a_4 = b_4 - b_3 - 1 = 0$$

$$\rightsquigarrow Z(1)$$

$$g^{-1}(100)$$

$$B^{-1}(1)$$

$$S(1)$$

$$S(1)$$

$$Z(1)$$

$$B^{-1}(0)$$

$$B^{-1}(0)$$

$$B^{-1}(1-1)$$

$$Z(1)$$

$$[\text{radif. alternativa: } 100+2 = 2 \cdot 3 \cdot 17 = P_2^1 \cdot P_2^2 \cdot P_3^0 \cdot P_4^0 \cdot P_5^0 \cdot P_6^0 \cdot P_7^1 = \begin{matrix} B^{-1}(0) \\ B^{-1}(1) \\ B^{-1}(0) \\ B^{-1}(0) \end{matrix}]$$

Chiaramente, una enumerazione dei programmi JRH induce una enumerazione delle funzioni calcolabili.

Def: Fissata una enumerazione effettiva $g: \mathbb{P} \rightarrow \mathbb{N}$ definiamo

1. $\varphi_m^{(k)}$ la funzione di k argomenti (k -aria) calcolata dal programma di codice m , ovvero da $P_m = g^{-1}(m)$

(con la notazione preced. introdotta $\varphi_m^{(k)} = f_{P_m}^{(k)}$)

2. $W_m^{(k)} = \text{dom}(\varphi_m^{(k)}) \subseteq \mathbb{N}^k$

3. $E_m^{(k)} = \text{cod}(\varphi_m^{(k)}) \subseteq \mathbb{N}$

se $K=1$ spesso viene omesso, e.g. scriviamo φ_m invece di $\varphi_m^{(1)}$

OSSERVAZIONE: La funzione

$\varphi^{(k)}: \mathbb{N} \rightarrow \mathcal{C}^{(k)}$ \leftarrow funzioni calcolabili con k argomenti
 $m \mapsto \varphi_m^{(k)}$

è ovviamente surgettiva (ogni funzione calcolabile è calcolata da qualche progr.)

$\mathcal{C}^{(k)}$ è numerabile

$$|\mathcal{C}^{(k)}| = |\mathbb{N}|$$

(in realtà dall'insieme di una funzione surgettiva $\mathbb{N} \rightarrow \mathcal{C}^{(k)}$ segue che

$|\mathcal{C}^{(k)}| < |\mathbb{N}|$, ma chiaramente esistono infiniti numeri calcolabili ad es. le costanti $1, \pi, \sqrt{2}, \dots$ e così via vale anche $|\mathcal{C}^{(k)}| \geq |\mathbb{N}|$)

Chiaro comunque $\varphi^{(k)}: \mathbb{N} \rightarrow \mathcal{C}^{(k)}$ non è iniettiva, anzi, dato che per ogni funzione calcolabile vi sono infiniti programmi che la calcolano

$$\forall f \in \mathcal{C}^{(k)}$$

$$|\varphi^{(k)}{}^{-1}(f)| = |\mathbb{N}|$$

ovvero

$$\varphi_0^{(k)} \quad \varphi_1^{(k)} \quad \varphi_2^{(k)} \quad \dots$$

è una enumerazione di $E^{(k)}$ con (infinte) ripetizioni.

Una enumerazione senza ripetizioni può essere definita

$$x(0) = 0$$

$$x(m+1) = \mu z. (\varphi_z \notin \{ \varphi_{x(0)}, \dots, \varphi_{x(m)} \})$$

\exists

$$\varphi_{x(0)} \quad \varphi_{x(1)} \quad \varphi_{x(2)} \quad \dots$$

ma questa è una enumerazione senza ripetizioni (molto) ineffettiva

(Si può dimostrare che esiste una $h: \mathbb{N} \rightarrow \mathbb{N}$ totale calcolabile t.c.

$\varphi_{h(0)} \varphi_{h(1)} \dots$ è una enum. senza ripetizioni [Friedberg 1958],
ma per noi va bene anche una enumerazione con ripetizioni.)

Teorema: La classe di tutte le funzioni calcolabili è enumerabile.
dim

$$C = \bigcup_{k \geq 1} E^{(k)}$$

unione numerabile di insiemi numerabili
→ è numerabile.

Nota: Ribadiamo che d'ora in poi supponiamo implicitamente fissata la numerazione effettiva dei programmi χ
ma il significato di $\varphi_m^{(k)}$, $W_m^{(k)}$, $E_m^{(k)}$ è fissato e determinato da tale enumerazione.

METODO DIAGONALE di CANTOR

La tecnica di diagonalizzazione, in senso generale, permette di costruire un oggetto che differisce da una infinità numerabile di altri oggetti.

z

Idea: dato un insieme numerabile di oggetti con struttura (funzioni, matrici, ...)
 $\{x_0, x_1, x_2, \dots\}$

si può costruire un oggetto x dello stesso tipo, ma diverso da tutti questi facendo in modo che " x differisce di x_m su m ".

Esempio: $|2^{\mathbb{N}}| > |\mathbb{N}|$

(Usa originale di Cantor, fondatore della teoria moderna degli insiemi per provare che ci sono vari "livelli di infinito".)

esistenza di insiemi infiniti era dubbia.

problemi di tipo religioso

dipendenze alla scoperta del paradosso di Russell

dim

Supponiamo, per assurdo, $|2^{\mathbb{N}}| = |\mathbb{N}|$, quindi esiste una enumeraz. di $2^{\mathbb{N}}$

$x_0 \ x_1 \ x_2 \ \dots$

consideriamo

N	2 ^N	x_0	x_1	x_2	\dots
0	?				
1	?				
2	?				
3	?				

e definiamo X che differisce da x_m su m

$$X = \{m \mid m \notin x_m\}$$

ovviamente $X \in 2^{\mathbb{N}}$ ma $\exists k$ t.c. $X = x_k$

ci chiediamo $k \in X$?

$$k \in X \Rightarrow k \notin x_k = X$$

assurdo!

$$k \notin X \Rightarrow k \in x_k = X$$

$\Rightarrow 2^{\mathbb{N}}$ non numerabile.

□

Corollario

se $|N \rightarrow N| = \{ \text{funzioni parziali da } N \text{ in } N \}$
allora

$$|N \rightarrow N| > |N|$$

dim

- E' sufficiente osservare che

$$|2^N| = |\{f: N \rightarrow \{0,1\} \mid f \text{ totale}\}| \leq |N \rightarrow N|$$

↑
funzioni caratteristiche

quindi

$$|N \rightarrow N| \geq |2^N| > |N|$$

- Alternativamente si può fare una dim diretta con la tecnica diagonale

sia f_0, f_1, f_2, \dots una enumerazione di elementi in $N \rightarrow N$
e si consideri

		IN → N		
		f_0	f_1	f_2
		0	1	1
0	-	-	1	-
1	-	-	-	1
2	-	-	-	-

possiamo definire una funzione f che differisce da tutte le f_m sulla diagonale.

$$f(m) = \begin{cases} 0 & \text{se } f_m(m) \uparrow \\ \uparrow & \text{se } f_m(m) \downarrow \end{cases} \in N \rightarrow N$$

allora $\forall m \quad f(m) \neq f_m(m) \Rightarrow f \neq f_m$

→ l'enumerazione non contiene tutte le funzioni in $N \rightarrow N$. \square

Nota: L'insieme $\bar{E} = \{f: N \rightarrow N \mid f \text{ non calcolabile}\}$

non è numerabile

dim

sappiamo che $|E| = |N|$. se \bar{E} fosse numerabile

allora $N \rightarrow N = E \cup \bar{E}$ sarebbe numerabile

unione di insiem numerabili è numerabile \square

Esercizio: Esiste una funzione totale non calcolabile

Gia' lo sapevamo per ragioni di cardinalita', ma ora siamo in grado di dimostrarlo.

$$f(m) = \begin{cases} \varphi_m(m) + 1 & \text{se } \varphi_m(m) \downarrow \quad (\text{ovvero } m \in W_m) \\ 0 & \text{se } \varphi_m(m) \uparrow \quad (" \quad m \notin W_m) \end{cases}$$

$\varphi_0 \quad \varphi_1 \quad \varphi_2$

0 $\varphi_0(0) \quad \varphi_1(0) \quad \varphi_2(0)$

1 $\varphi_0(1) \quad \varphi_1(1) \quad \varphi_2(1)$

2 $\varphi_0(2) \quad \varphi_1(2) \quad \varphi_2(2)$

è facile vedere che

- f totale
- $\forall m \quad f \neq \varphi_m \Rightarrow f$ non calcolabile.

Note: Sono infiniti le funzioni totali non calcolabili

$$f(m) = \begin{cases} \varphi_m(m) + k & m \in W_m \\ k & m \notin W_m \end{cases}$$

Quante sono?

Esercizio: Sia $f: \mathbb{N} \rightarrow \mathbb{N}$ (totale), $m \in \mathbb{N}$

Definire una funzione $g: \mathbb{N} \rightarrow \mathbb{N}$ non calcolabile. E.c.

$$g(x) = f(x) \quad \forall x < m$$

Si può utilizzare una "diagonale trasposta"

	φ_0	φ_1	φ_2
0			
1			
\vdots			
m			
$m+1$			
$m+2$			

$$g(x) = \begin{cases} f(x) & x < m \\ \varphi_{x-m}(x) + 1 & x \geq m \text{ e } x \in W_{x-m} \\ 0 & x \geq m \text{ e } x \notin W_{x-m} \end{cases}$$

chiaramente $\forall x \quad g \neq \varphi_x$ in quanto $g(x+m) \neq \varphi_x(x+m)$

Nota: Si poteva anche definire

$$g(x) = \begin{cases} f(x) & x < m \\ \varphi_x(x) + 1 & x \geq m \text{ e } x \in W_x \\ 0 & x \geq m \text{ e } x \notin W_x \end{cases}$$

purché? Tanto ogni funzione compare infinite volte nell'enumerazione non saltare i primi $m-1$ indici non crea problemi.

formalmente, sappiamo che $\forall x \geq m \quad g \neq \varphi_x$

dunque $\forall y$ dato che ci sono infiniti indici per φ_y

$$\exists x \geq m \text{ tc } \varphi_y = \varphi_x \text{ e } \varphi_x \neq g$$

$\forall y \quad \varphi_y \neq g \leadsto g$ non calcolabile!

Esercizio: Data una famiglia di funzioni $\{f_i\}_{i \in \mathbb{N}}$ $f_i : \mathbb{N} \rightarrow \mathbb{N}$

definire $g : \mathbb{N} \rightarrow \mathbb{N}$ tc $\text{dom}(g) \neq \text{dom}(f_i) \quad \forall i \in \mathbb{N}$

$$g(m) = \begin{cases} 0 & \text{se } m \notin \text{dom}(f_m) \\ \uparrow & \text{se } m \in \text{dom}(f_m) \end{cases}$$

in questo modo $\forall m$.

$$\text{se } m \in \text{dom}(f_m) \quad \text{allora } g(m) \neq f_m(m)$$

Esercizio: Definire una funzione non calcolabile totale che vale 1 sui numeri pari

$$f(x) = \begin{cases} 1 & x \text{ pari} \\ \frac{\varphi_{x-1}(x)+1}{2} & x \text{ dispari e } x \in W_{\frac{x-1}{2}} \\ 0 & x \text{ dispari e } x \notin W_{\frac{x-1}{2}} \end{cases}$$

E' facile dimostrare che la funzione f non è calcolabile

Infatti $\forall m$

$$f(zm+1) \neq \varphi_m(zm+1)$$

se

$$2m+1 \in W_m \Rightarrow f(2m+1) = \varphi_m(2m+1) + 1 \neq \varphi_m(2m+1)$$

se

$$2m+1 \notin W_m \Rightarrow f(2m+1) = 0 \neq \varphi_m(2m+1)$$

TEOREMA di PARAMETRIZZAZIONE S-M-N (Kleene)

Intuizione: Sia $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ una funzione calcolabile.

→ esiste certamente un indice e (in realtà ne esistono infiniti) tale che

$$f(x,y) = \varphi_e^{(2)}(x,y)$$

ora, se fissiamo il primo argomento ad un certo valore x , si ottiene una funzione di un solo argomento $f_x: \mathbb{N} \rightarrow \mathbb{N}$

$$f_x(y) = \varphi_e^{(2)}(x,y)$$

e $\forall x \in \mathbb{N} f_x$ è calcolabile (composizione di funzioni calcolabili), dunque esiste $d \in \mathbb{N}$ te

$$f_x = \varphi_d$$

ovvero $\forall y \in \mathbb{N}$

$$f_x(y) = \varphi_e^{(2)}(x,y) = \varphi_d(y)$$

e chiaramente d dipende da e e x → $d = s(e,x)$

com. $s: \mathbb{N}^2 \rightarrow \mathbb{N}$

funz. totale

Il teorema S-m-n ci dice che la funzione s è calcolabile!

intuitivo: come posso calcolare $s(e,x)$

- da e determino il programma $P_e = \gamma^{-1}(e)$ che calcola $\varphi_e^{(2)}(x,y) = f(x,y)$

- il programma che calcola $f_x = \lambda y. f(x,y)$ com. x fissato si può facilmente ottenere da P_e

- sposta y nel reg. 2 }
- mette x " reg. 1 }
- esegue P_e

- si prende il codice del programma ottenuto

3c Ancora intuitivamente:

- funzioni su indici, come s , sono funzioni che trasformano programmi
- il teorema S-m-m ci dice che l'operazione di fissare un argomento di un programma è effettiva.

Esempio: si consideri la funzione calcolabile

$$f(x,y) = xy$$

dove esiste un indice e t.c. $\varphi_e = f$, ovvero

$$\varphi_e(x,y) = f(x,y) = xy$$

Quindi al variare di x otengo le funzioni calcolabili

$$f_0(y) = y^0 = 1 \quad \rightsquigarrow \text{indice } s(e, 0)$$

$$f_1(y) = y^1 = y \quad \rightsquigarrow s(e, 1)$$

$$f_2(y) = y^2 = y \quad \rightsquigarrow s(e, 2)$$

per il teorema, questi indici si possono determinare in modo effettivo

Il teorema si formula, in generale, per funzioni $f(\vec{x}, \vec{y}) : \mathbb{N}^{m+n} \rightarrow \mathbb{N}$ di cui il nome

Teorema S-m-m (Kleene)

Dati $m, n \geq 1$ esiste una funzione totale calcolabile

$$s_{m,n} : \mathbb{N}^{m+1} \rightarrow \mathbb{N}$$

tale che

$$\varphi_e^{(m+n)}(\vec{x}, \vec{y}) = \varphi_{s_{m,n}(e, \vec{x})}^{(n)}(\vec{y}) \quad \forall e \in \mathbb{N} \\ x \in \mathbb{N}^m, y \in \mathbb{N}^n$$

dim

intuitivamente, dati $e \in \mathbb{N}$, $\vec{x} \in \mathbb{N}^m$ si può:

- tramite γ^{-1} ottenere il programma $P_e = \gamma^{-1}(e)$ che calcola (standard)

la funzione $q_{e^{(m+m)}}$, ovvero t.c partendo da

\vec{x}	\vec{y}	10 0 ...
m	m	

calcolo $q_{e^{(m+m)}}(\vec{x}, \vec{y})$

- a partire da Pe possiamo chiaramente costruire un nuovo programma P che partendo da

\vec{y}	10 0 ...
-----------	----------

calcolo $q_{e^{(m+m)}}(\vec{x}, \vec{y})$

Infatti, è sufficiente

- spostare \vec{y} "in avanti" di m registri
- caricare \vec{x} negli m registri così liberati
- eseguire Pe

Il programma P può essere

$$T([1, m], [m+1, m+m])$$

$Z(1)$

$$\begin{cases} S(1) \\ x_1 \text{ volte} \end{cases}$$

$S(1)$

$Z(m)$

$$\begin{cases} S(m) \\ x_m \text{ volte} \end{cases}$$

$S(m)$

Pe

☞ (si ricordi che la costruzione deve aggiornare tutte le istruzioni di salto in Pe)

$$J(m', m', t) \rightarrow J(m', m', t + m + m + \sum_{i=1}^m x_i)$$

una volta costruito P si può porre

$$S(e, \vec{x}) = f(P).$$

Ogni funzione e costruzione utilizzata è effettiva. (in particolare lo sono f, f^{-1})

questo dimostra (omiché informalmente, ricorrendo alla tesi di Church) esistenza, totalità e calcolabilità di S

La dimostrazione di calcolabilità formale è lunga, ma non difficile...
basta procedere per gradi, seguendo l'intuizione della dimostrazione informale.

① funzione di approssimamento di un programma

$$\text{agg} : \mathbb{N}^2 \rightarrow \mathbb{N}$$

$\text{agg}(e, t) =$ codice di un programma ottenuto da $P_e = f^{-1}(e)$
sommando t allo offset massimo di ogni istruzione
di salto

per questo è opportuno definire una funzione di supporto che opera sulla
singola istruzione (codificata con β)

$$\tilde{\text{agg}} : \mathbb{N}^2 \rightarrow \mathbb{N}$$

$\tilde{\text{agg}}(i, t) =$ codice dell'istruzione ottenuta approssimando $\beta^{-1}(i)$
- se è $J(-, -, -) \approx J(2, -, - + t)$
- altrimenti invariata

dati $i, t \in \mathbb{N}$ e detto $q = qt(4, i)$ $\tau = \text{rem}(4, i)$

vale

$$\tilde{\text{agg}}(i, t) = \begin{cases} 4 * \sqrt{(\nu_1(q), \nu_2(q), \nu_3(q) + t)} + 3 & \tau = 3 \\ i & \tau \neq 3 \end{cases}$$

$\tilde{\text{agg}}$

computabile (definizione per costruzione e composizione) anzi è PPR

a questo punto:

$$\text{agg}(e, t) = \tau(\tilde{\text{agg}}(a(e, 1), t) \dots \tilde{\text{agg}}(a(e, l(e)), t))$$

$$= \sum_{i=1}^{l(e)} 2 \left(\sum_{j=1}^i \tilde{\text{agg}}(a(e, j), t) + (i-1) \right)$$

(usa la codifica del libro, per le funzioni)
con la codifica alternativa nel 35 bis

② concatenazione di programmi

$$\text{seq} : \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$\text{seq}(e_1, e_2) = \gamma \left(\frac{P_{e_1}}{P_{e_2}} \right) = \gamma(P_{e_1} \text{ Pag}(e_2, P_{e_1}))$$

è opportuno definire preliminarmente una funzione per lo concatenazione di sequenze

$$c: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$c(e_1, e_2) = \tau(a(e_1, 1) \dots a(e_1, l(e_1)), a(e_2, 1) \dots a(e_2, l(e_2)))$$

$$= \underbrace{\sum_{i=1}^{l(e_1)} 2 \sum_{j=1}^i a(e_1, j) + (i-1)}_{e_1} + \underbrace{\sum_{i=1}^{l(e_2)} 2 \sum_{j=1}^{l(e_1)} a(e_2, j) + \sum_{j=1}^i a(e_2, j) + l(e_2) + i - 1}_{b(e_1, e_2, i)}$$

$$= e_1 + \underbrace{\sum_{i=1}^{l(e_2)} 2 b(e_1, l(e_1)) + \sum_{j=1}^i a(e_2, j) + i}_{e_2}$$

a questo punto

$$\text{seq}(e_1, e_2) = c(e_1, \text{agg}(e_2, l(e_2))) \in PR$$

(3) sottoprogramma per il trasferimento di $[1..m] \rightsquigarrow [m+1..m+m]$

$$\text{trasf}: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$\text{trasf}(m, m) = \gamma(\tau([1..m], [m+1..m+m]))$$

$$= \tau(\beta(\tau(1, m+1)) \dots \beta(\tau(m, m+m)))$$

$$= \sum_{i=1}^m 2 \sum_{j=1}^i \beta(\tau(j, m+j)) + (i-1)$$

$$= \sum_{i=1}^m 2 \left(\sum_{j=1}^i (4 * \pi(j-1, m+j-1) + 2) \right) + (i-1)$$

$$\in PR$$

(4) sottoprogramma che setta $R_i = xi$

$$\text{set}: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$\text{set}(i, x) = \gamma \left(\begin{matrix} z(i) \\ s(i) \\ \vdots \\ s(i) \end{matrix} \right) \xrightarrow{x \text{ volte}} =$$

$$= \tau(\beta(z(i)) \dots \beta(s(i)) \dots \beta(s(i)))$$

$$= 2 \beta(z(i)) + \sum_{k=1}^x 2 \sum_{j=1}^k \beta(s(j)) + (k+1)-1$$

$$= 2^{4*(i-1)} + \sum_{k=1}^x 2^{k*(4*(i-1)+1) + k}$$

$$\in PR$$

* Funzioni per il teorema SHN nello codice alternativo

$$\textcircled{1} \quad \text{agg}(e, t) = \tau(\tilde{\text{agg}}(\alpha(e, 1), t) \dots \tilde{\text{agg}}(\alpha(e, l(e)), t))$$

$$= \left(\prod_{i=1}^{l(e)-1} p_i^{\tilde{\text{agg}}(\alpha(e, i), t)} \right) p_{l(e)}^{\tilde{\text{agg}}(\alpha(e, l(e))) + 1} - 2$$

$$\textcircled{2} \quad \text{seq}(e_1, e_2) = c(e_1, \text{agg}(e_2, l(e_1)))$$

dove

$$c(e_1, e_2) = \tau(\alpha(e_1, 1) \dots \alpha(e_1, l(e_1)) \alpha(e_2, 1) \dots \alpha(e_2, l(e_2)))$$

$$= \left(\prod_{i=1}^{l(e_1)} p_i^{\alpha(e_1, i)} \right) \left(\prod_{i=1}^{l(e_2)-1} p_i^{\alpha(e_2, i)} \right) p_{l(e_1)+l(e_2)}^{\alpha(e_2, l(e_2)) + 1} - 2$$

$$\textcircled{3} \quad \text{trasf}(m, m) = \gamma(T(1, m+1) \dots T(m, m+m))$$

$$= \tau(\beta(T(1, m+1)) \dots \beta(T(m, m+m)))$$

$$= \left(\prod_{i=1}^{m-1} p_i^{\beta(T(i, m+i))} \right) p_m^{\beta(T(m, m+m)) + 1} - 2$$

$$= \left(\prod_{i=1}^{m-1} p_i^{4 \times \pi(i-1, m+i-1) + 2} \right) p_m^{4 \times \pi(m-1, m+m-1) + 1} - 2$$

$$\textcircled{4} \quad \text{set}(i, x) = \gamma \begin{pmatrix} z(i) \\ s(i) \\ s(i) \end{pmatrix}_{\text{svolti}}$$

$$= \tau(\beta(z(i)) \beta(s(i)) \dots \beta(s(i))) =$$

$$= p_2^{\beta(z(i))} \cdot \prod_{i=1}^{x-1} p_{i+1}^{\beta(s(i))} \cdot p_{x+1}^{\beta(s(i)) + 1} - 2$$

$$= p_2^{4 \times (i-1)} \cdot \prod_{i=1}^{x-1} p_{i+1}^{4 \times (i-1) + 2} \cdot p_{x+1}^{4 \times (i-1) + 2} - 2$$

(36)

A questo punto, indicato con

$$\text{pref}_{m,m} : \mathbb{N}^m \rightarrow \mathbb{N}$$

$$\text{pref}_{m,m}(\vec{x}) = \text{seq}(\text{transf}(m,m), \text{seq}(\text{set}(1,x_1), \text{seq}(\dots, \text{set}(m,x_m))))$$

si ha che

$$s_{m,m} : \mathbb{N}^{m+1} \rightarrow \mathbb{N}$$

$$s_{m,m}(e, \vec{x}) = \text{seq}(\text{pref}_{m,m}(\vec{x}), e) \quad \in PR$$

□

Nota: La prova dimostra che $s_{m,m}$ non solo è calcolabile totale, ma è più ovvia ricorsiva.

Il teorema viene spesso utilizzato in una forma semplificata:

Corollario: Sia $f : \mathbb{N}^{m+m} \rightarrow \mathbb{N}$ una funzione calcolabile.

Allora esiste una funzione calcolabile totale $s : \mathbb{N}^m \rightarrow \mathbb{N}$ t.c.

$$\forall x \in \mathbb{N}^m$$

$$\forall y \in \mathbb{N}^m \quad f(\vec{x}, \vec{y}) = \varphi_{s(\vec{x})}(\vec{y})$$

dim

se f calcolabile $\Rightarrow f = \varphi_e^{(m+m)}$ per qualche $e \in \mathbb{N}$

$$\text{ma } s(\vec{x}) = s_{m,m}(e, \vec{x})$$

□

Esercizio 1: Dimostrare che esiste una funzione totale calcolabile $K : \mathbb{N} \rightarrow \mathbb{N}$

t.c.

$$\forall m \quad \varphi_{K(m)} = \lambda x. \lfloor \sqrt[m]{x} \rfloor$$

ovvero K è una enumerazione delle funzioni $\lfloor \sqrt[m]{x} \rfloor$ (un po' più di questi)

K è una funzione che dato m vi restituisce il programma
che calcola $\lfloor \sqrt[m]{x} \rfloor$

Definiamo

$$f(m, x) = \lfloor \sqrt[m]{x} \rfloor \equiv \exists y \leq x \quad (y+1)^m > x$$

$$= \exists y \leq x \quad (x+1 \leq (y+1)^m)$$

calcolabile!

La funzione f è calcolabile, dunque per il Corollario al teorema S-m-m esiste una funzione calcolabile totale $K: \mathbb{N} \rightarrow \mathbb{N}$ t.c.

$$\varphi_{K(m)}(x) = f(m, x) = \lfloor \sqrt[m]{x} \rfloor$$

come desiderato.

Esercizio 2 : Dimostrare che esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ totale calcolabile t.c.

$\forall m$. $\varphi_{K(m)}$ è definita solo sulle potenze m -ime

ovvero

$$W_{K(m)} = \{x \mid \exists y \in \mathbb{N} \quad x = y^m\}$$

definiamo

$$f: \mathbb{N}^2 \rightarrow \mathbb{N}$$

$$f(m, x) = \begin{cases} 0 & \text{se } x \text{ potenza } m^{\text{ma}} \\ \uparrow & \text{altrimenti} \end{cases}$$

$$= \Omega(\mu y. \mid y^m - x \mid)$$

ma andava anche bene

$$f(m, x) = \begin{cases} \sqrt[m]{x} & \text{se } x \text{ potenza } m^{\text{ma}} \\ \uparrow & \text{altrimenti} \end{cases}$$

$$= \mu y. \mid y^m - x \mid$$

calcolabile n. x il teorema SMM esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale tali che $\forall m$

$$f(m, x) = \varphi_{K(m)}(x)$$

↓
K soddisfa la richiesta... infatti

$$x \in W_{K(m)} \iff \varphi_{K(m)}(x) \downarrow \iff f(m, x) \downarrow \iff x \text{ potenza } m^{\text{ma}}$$

Nota. K è una enumerazione dei sottosinsiemi di naturali "potenza m^{ma} "

Esercizio 3: Si dimostri che esiste una funzione $s: \mathbb{N} \rightarrow \mathbb{N}$ totale calc.

t.c.

$$\mathbb{W}_{s(x)}^{(k)} = \{(y_1, \dots, y_k) \mid y_1 + \dots + y_k = x\}$$

Definiamo

$$f(x, \vec{y}) = \begin{cases} 0 & \text{se } y_1 + \dots + y_k = x \\ \uparrow & \text{altrimenti} \end{cases}$$

$$= \mu z. \exists \vec{y}. \vec{y} \in \mathbb{W}_{s(x)}^{(k)}$$

\exists

x th. s.m.m. esiste $s: \mathbb{N} \rightarrow \mathbb{N}$ totale calc t.c.

$$f(x, \vec{y}) = \varphi_{s(x)}^{(k)}(\vec{y})$$

\exists

$$\vec{y} \in \mathbb{W}_{s(x)}^{(k)} \text{ ose } \varphi_{s(x)}^{(k)}(\vec{y}) \downarrow \text{ ose}$$

$$\text{o ose } f(x, \vec{y}) \downarrow \text{ ose } y_1 + \dots + y_k = x$$

ovvero

$$\mathbb{W}_{s(x)}^{(k)} = \{(y_1, \dots, y_k) \mid y_1 + \dots + y_k = x\}$$

come desiderato. \square

FUNZIONE UNIVERSALE

Vedremo ora come la teoria sviluppata finora consente di provare un risultato in qualche senso sorprendente, ovvero l'esistenza di funzioni / programmi universali, in grado di riprodurre il comportamento di ogni altro funzione calcolabile / programma.

{ teorema s-m-m
funzioni / programmi universali } → pilastri della teoria della calcolabilità

Si consideri la funzione

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

in un certo senso ψ è una funzione "universale", che contiene tutte le funzioni calcolabili umane

$$\varphi_0, \varphi_1, \varphi_2, \dots$$

infatti, per ogni $e \in \mathbb{N}$ fissato

$$g(y) = \psi(e, y) = \varphi_e(y) \quad \rightsquigarrow g = \varphi_e$$

cioè al variare di e , ψ rappresenta ogni funz. calc. umana.

Più in generale

Def. La funzione universale per le funzioni k -arie ($k \in \mathbb{N}$) è definita da

$$\psi^{(k)}: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$$

$$\psi^{(k)}(e, \vec{x}) = \varphi_e(\vec{x})$$

Problema: Le φ_e sono funzioni bene definite, ma sono calcolabili?

In caso affermativo vi sarebbe un programma P_0 in grado di calcolare tutte le funzioni calcolabili k -arie

→ include in sé ogni possibile altro programma...

calcolatore universale [M. Davis "Il calcolatore universale

Da Leibniz a Turing"]

A prima vista può sembrare strano, ma se ci pensiamo un po' meglio...

un programma universale

- riceve un input \bar{x} indice e (descrizione del programma) Pe
ob. eseguire
 - * argomenti \bar{x}
 - ed esegue Pe sugli argomenti!
- Σ
- è un interprete!!

Ed in effetti vale il seguente risultato

Teorema: $\forall k \geq 1$ la funzione universale $\psi^{(k)}_0$ è calcolabile

dim

* Informale: Sia $k \geq 1$ fissato;
dati un indice $e \in \mathbb{N}$ e gli argomenti $\bar{x} \in \mathbb{N}^k$
si può calcolare $\psi^{(k)}_0(e, \bar{x}) = \varphi_e^{(k)}(\bar{x})$ come segue

- si costruisce il programma $Pe = \gamma^{-1}(e)$
- si simula Pe sull'input \bar{x}
- se $Pe(\bar{x}) \downarrow$, il valore di $\psi^{(k)}_0(e, \bar{x})$ è in R_1
altun. ok

tutto è effettivo \Rightarrow per la tesi di Church $\psi^{(k)}_0$ è calcolabile

... insoddisfacente, questo è un risultato cruciale per la teoria!

* Una dimostrazione formale mostra che, mediante funzioni calcolabili,
è possibile simulare i simboli possibili di ogni macchina data ... come già
fatto nella dimostrazione $R = C$

Più precisamente:

- si codifica la configurazione dei registri di una macchina, dove
solo un numero finito di registri è $\neq 0$ con un numero

r_1	r_2	r_3	\dots
-------	-------	-------	---------

$$C = \prod_{i \geq 1} p_i^{n_i}$$

così che il valore dei registri

$$\forall i \quad r_i = (c)_i$$

- lo stato della macchina, ad ogni istante, si codifica come

$$\sigma = \pi(j, c)$$

prox istruz. da eseguire config. dei registri

L'idea è dunque quello di mostrare che sono calcolabili (anzi, $\in PR$) le funzioni.

$$c_k : \mathbb{N}^{K+2} \rightarrow \mathbb{N}$$

$$c_k(e, \vec{x}, t) = \begin{cases} \text{config. dei registri dopo } t \text{ passi di } Pe(\vec{x}), \text{ se la comp. mom.} \\ \text{termina in memo di } t \text{ passi} \\ \text{conf. finale dei registri, altrimenti} \end{cases}$$

$$j_k : \mathbb{N}^{K+2} \rightarrow \mathbb{N}$$

$$j_k(e, \vec{x}, t) = \begin{cases} \text{numero dell'istruz. da eseguire al } t+1\text{-mo passo di } Pe(\vec{x}) \\ \text{se la comp. mom. termina in } t \text{ passi o memo} \\ \text{o altrimenti} \end{cases}$$

A questo punto

$$\varphi_0^{(k)}(e, \vec{x}) = \varphi_e^{(k)}(\vec{x}) = (c_k(e, \vec{x}, \mu t), j_k(e, \vec{x}, t))_1$$

dunque, se dimostriamo che c_k, j_k calcolabili potremmo concludere che anche $\varphi_0^{(k)}$ è calcolabile

Si procede come nella dim. di $R = e$ (anzi questo può essere vista come una dim. più formale dello stesso fatto), dimostrando che $c_k, j_k \in PR$

l'unica differenza: nella dim. precedente definivamo C_p e J_p con P fissato mentre invece ora P è un parameetro

$$C_p(\vec{x}, t) = \begin{cases} \dots & \text{se l'istruzione } J_p(\vec{x}, t) \text{ di P è} \\ & \text{e deve diventare} \end{cases}$$

$$c_k(e, \vec{x}, t) = \begin{cases} \dots & \text{se l'istruzione } J_k(e, \vec{x}, t) \text{ di Pe è} \\ & \text{facilmente esprimibile in PR} \end{cases}$$

Explicitamente, per possi:

(a) argomenti di una istruzione URM ($i = \beta(\text{istr})$)

$$Z_{\text{arg}}(i) = qt(4, i) + 1$$

$$S_{\text{arg}}(i) = qt(4, i) + 1$$

$$T_{\text{arg}_h}(i) = \pi_h(qt(4, i)) + 1 \quad h \in \{1, 2\}$$

$$J_{\text{arg}_h}(i) = \vartheta_h(qt(4, i)) + 1 \quad h \in \{1, 2, 3\}$$

(b) effetto dell'esecuzione di una istruzione sulla configurazione c

$$\text{zero}(c, m) = qt(p_m^{(c)_m}, c) \quad z(m)$$

$$\text{succ}(c, m) = p_m \cdot c \quad s(m)$$

$$tfc(c, m, m) = qt(p_m^{(c)_m}, c) * p_m^{(c)_m} \quad T(m, m)$$

(c) effetto sulla configurazione dei registri dall'esecuzione dell'istruz. $i = \beta(\text{istr})$

$$\text{change}(c, i) = \begin{cases} \text{zero}(c, Z_{\text{arg}}(i)) & \text{rm}(4, i) = 0 \\ \text{succ}(c, S_{\text{arg}}(i)) & \text{rm}(4, i) = 1 \\ tfc(c, T_{\text{arg}1}(i), T_{\text{arg}2}(i)) & \text{rm}(4, i) = 2 \\ c & \text{rm}(4, i) = 3 \end{cases}$$

(d) conf. dei registri se quella attuale c viene eseguita la t-ma istruz. di Pe

$$\text{next conf}(e, c, t) = \begin{cases} \text{change}(c, a(e, t)) & \text{se } 1 \leq t \leq l(e) \\ c & \text{altrimenti} \end{cases}$$

per il cambiamento del program counter

(e) num. dell'istruzione successiva, se viene eseguito l'istruzione $i = \beta(\text{istr})$ t-ma istruzione del programma

$$is(c, i, t) = \begin{cases} t+1 & \text{se } (\text{rm}(4, i) \neq 3) \circ (\text{rm}(4, i) = 3 \text{ e } (c)_{\text{arg}_1(i)} \neq (c)_{\text{arg}_2(i)}) \\ T_{\text{arg}_3}(i) & \text{altrimenti} \\ & \text{se } \text{rm}(4, i) = 3 \text{ e } ((c)_{\text{arg}_1(i)} = (c)_{\text{arg}_2(i)}) \end{cases}$$

(f) num. istruzione successiva se si esegue la t-ma istruz. di Pe su conf. c

$$\text{next inst}(e, c, t) = \begin{cases} is(c, a(e, t), t) & \text{se } 1 \leq t \leq l(e) \\ is(c, a(e, t), t) \leq l(e) & \\ 0 & \text{altrimenti} \end{cases}$$

A questo punto possiamo definire C_K e J_K

$$C_K(e, \vec{x}, 0) = \prod_{i=1}^k p_i^{x_i}$$

$$J_K(e, \vec{x}, 0) = 1$$

$$C_K(e, \vec{x}, t+1) = \text{mext_comf}(e, C_K(e, \vec{x}, t), J_K(e, \vec{x}, t))$$

$$J_K(e, \vec{x}, t+1) = \text{mext_imst}(e, C_K(e, \vec{x}, t), J_K(e, \vec{x}, t))$$

$$\Theta_K(e, \vec{x}, t) = \Pi(J_K(e, \vec{x}, t), C_K(e, \vec{x}, t))$$

può essere definita per ricorrenza primitiva $\sim \in PR$

C_K, J_K sono in PR

$$\psi_J^{(K)}(e, \vec{x}) = C_K(e, \vec{x}, \mu t. J_K(e, \vec{x}, t)) \quad e \in R = C$$



□

Come semplice corollario otteniamo la decidibilità di due predicati che saremo molto utili nel seguito

Corollario: I seguenti predicati sono decidibili

$$(a) H_K(e, \vec{x}, t) \equiv "P_e(\vec{x}) \downarrow \text{in } t \text{ o meno passi}"$$

$$(b) S_K(e, \vec{x}, y, t) \equiv "P_e(\vec{x}) \downarrow y \text{ in } t \text{ o meno passi}"$$

dim

(a) basta osservare che

$$H_K(e, \vec{x}, t) \equiv (J_K(e, \vec{x}, t) = 0)$$

$$\chi_{H_K}(e, \vec{x}, t) = \overline{\text{sg}}(J_K(e, \vec{x}, t))$$

(b) si osserva che

$$S_K(e, \vec{x}, y, t) \equiv (J_K(e, \vec{x}, t) = 0) \wedge ((C_K(e, \vec{x}, t))_1 = y))$$

□

Inoltre deduciamo dal teorema la possibilità di esprimere ogni funzione calcolabile nella cosiddetta forma normale di Kleene.

Corollario (forma normale di Kleene)

$\forall e, k \in \mathbb{N} \quad \forall x \in \mathbb{N}^k$

$$q_e^{(k)}(\bar{x}) = (\mu z. "S_k(e, \bar{x}, (z)_1, (z)_2)")_1$$

$$= (\mu z. |x_{S_k(e, \bar{x}, (z)_1, (z)_2)} - 1|)_1$$

3

OSS:

- Il corollario evidenzia come ogni funzione calcolabile (o equivalente parziale ricorsiva) possa essere ottenuta da funzioni primitive ricordive utilizzando la minimizzazione al più una volta.

[basta un solo while!]

- La minimizzazione permette di "cercare" un simbolo valore che gode di una certa proprietà.

quella utilizzata è una tecnica tipica per cercare coppie di valori generalizzabili a tuple.

* Applicazioni della funzione universale

Ricordiamo che abbiamo già osservato che se $f: \mathbb{N} \rightarrow \mathbb{N}$ è una funzione calcolabile totale imiettiva \rightarrow

$$f^{-1}(x) = \begin{cases} y & \text{se esiste } y \text{ tc. } f(y) = x \\ \uparrow & \text{altrimenti} \end{cases}$$

è calcolabile, in quanto $f^{-1}(x) = \mu y \cdot |f(y) - x|$

Verifichiamo che l'ipotesi di totalità non serve.

Teorema: Sia $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile imiettiva. Allora $f^{-1}: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile

dim

Dato che f è calcolabile, esiste $e \in \mathbb{N}$ tc. $\varphi_e = f$

Ora, basta osservare che

$$\begin{aligned} f^{-1}(x) &= (\mu z \cdot "S(e, (z)_1, x, (z)_2)")_1 \\ &= (\mu z \cdot |X_s(e, (z)_1, x, (z)_2) - 1|)_1 \\ &\quad \uparrow \text{usiamo omot. } S \end{aligned}$$

Note: non era corretto

$$f^{-1}(x) = \mu y \cdot (\mu t \cdot |S(e, y, x, t) - 1|)$$

Inoltre possiamo individuare nuove funzioni non calcolabili e prediciuti non decidibili.

Teorema: Il problema " φ_x è totale" non è decidibile.

dim

sia $Tot(x)$ il predicato

$$Tot(x) \equiv "\varphi_x \text{ è totale}"$$

Supponiamo p. assurdo che sia decidibile

\exists

la funzione caratteristica

$$\chi_{Tot}(x) = \begin{cases} 1 & \varphi_x \text{ totale} \\ 0 & \text{altrimenti} \end{cases}$$

è calcolabile.

Allora la funzione

$$g(x) = \begin{cases} \varphi_x(x) + 1 & \text{se } \varphi_x \text{ totale} \\ 1 & \text{altrimenti} \end{cases}$$

è

- totale

- calcolabile (dato che " φ_x totale" decidibile e

$$\varphi_x(x) + 1 = \psi_0(x, x) + 1 \text{ calcolabile}$$

si conclude per lo dif per così

z

falso! richiedeva che le funzioni usate nei vari casi
fossero totali!

$$g(x) \neq (\psi_0(x, x) + 1) \cdot \chi_{\text{tot}}(x) + 0 \cdot (1 - \chi_{\text{tot}}(x))$$

imdefinita se $\psi_0(x, x) \uparrow$

z

Il teorema di definizione per così continua a valere per funz.
mom totali, ma la dim. deve essere cambiata (esercizio con
2 casi)

nello specifico:

$$\begin{aligned} g(x) &= (\mu z \cdot "s(x, x, (z)_1, (z)_2) \vee \neg \text{Tot}(x)")_1 + 1 \\ &= (\mu z \cdot |\chi_{s(x, x, (z)_1, (z)_2) \vee \neg \text{Tot}(x)} - 1|)_1 + 1 \end{aligned}$$

$$\text{e } \forall x \quad \text{se } \varphi_x \text{ totale} \Rightarrow g(x) = \varphi_x(x) + 1 \Rightarrow \varphi_x \neq g$$

ovvero g è calcolabile totale, ma diversa da ogni funzione calcolabile
totale no assurdo!

$\rightsquigarrow \text{Tot}(x)$ mom decidibile!

□

OSSERVAZIONE: allo stesso modo si può dimostrare che mom sono decidibili

- $P_1(x) \equiv "x \in W_x" \equiv "\varphi_x(x) \downarrow"$

HALTING PROBLEM

- $P_2(x, y) \equiv "y \in W_x" \equiv "\varphi_x(y) \downarrow"$

(è chiamato uno dei tanti...)

* Operazioni effettive su funzioni calcolabili

L'esistenza della funzione universale, insieme con il teorema SMM, ci permettono di dimostrare formalmente l'effettività di varie operazioni su (indici di) funzioni calcolabili.
su programmi!

Esempio - prodotto: dati x, y trovare w t.c. $\varphi_w = \varphi_x \cdot \varphi_y$
 $(\varphi_x \cdot \varphi_y)(z) = \varphi_x(z) \cdot \varphi_y(z)$

- inversa: dato x trovare w t.c. $\varphi_w = \varphi_x^{-1}$

è più o meno intuitivo che si tratti di manipolazioni effettive di programmi ma non ovvio come possano essere dimostrate formalmente calcolabili.

1) Prodotto

Esiste una funzione $s: \mathbb{N}^2 \rightarrow \mathbb{N}$ totale calcolabile t.c.

$$\varphi_s(x, y) = \varphi_x \cdot \varphi_y$$

dim

definiamo una funzione dove x, y sono argomenti... poi li trasf in parametri

$$g(x, y, z) = \varphi_x(z) \cdot \varphi_y(z)$$

$$= \psi_0(x, z) \cdot \psi_0(y, z)$$

\Rightarrow calcolabile (per la calcolabilità di ψ_0 , prodotto, composizione)

E

per il teorema SMM esiste $s: \mathbb{N}^2 \rightarrow \mathbb{N}$ t.c.

$$\varphi_{s(x, y)}(z) = g(x, y, z) = \varphi_x(z) \cdot \varphi_y(z)$$

ovvero

$$\varphi_{s(x, y)} = \varphi_x \cdot \varphi_y \quad \text{come desiderato}$$

2) Quadrato

Essiste $K: \mathbb{N} \rightarrow \mathbb{N}$ totale calcolabile t.c. $\varphi_{K(x)} = \varphi_x^2$

(ovvero $\forall z \quad \varphi_{K(x)}(z) = (\varphi_x(z))^2$)

dim

$$K(x) = s(x, \infty)$$

funzione del punto 1

3) Effettività della ricorsione

Ricordiamo lo schema di ricorsione primitiva

$$h(\vec{x}, 0) = f(\vec{x})$$

$$h(\vec{x}, y+1) = g(\vec{x}, y, h(\vec{x}, y))$$

e supponiamo che se f, g calcolabili $\Rightarrow h$ calcolabile.

Se

se $f = \varphi_{e_1}^{(k)}$ e $g = \varphi_{e_2}^{(k+2)}$ esiste un indice $r(e_1, e_2)$ t.c.

$$h = \varphi_{r(e_1, e_2)}^{(k+1)}$$

vogliamo dimostrare che $r: \mathbb{N}^2 \rightarrow \mathbb{N}$ è totale calcolabile

ovvero esiste $r: \mathbb{N}^2 \rightarrow \mathbb{N}$ totale calcolabile t.c. $\forall e_1, e_2$, se def.

$$h(\vec{x}, 0) = \varphi_{e_1}^{(k)}(\vec{x})$$

$$h(\vec{x}, y+1) = \varphi_{e_2}^{(k+2)}(\vec{x}, y, h(\vec{x}, y))$$

Allora

$$h = \varphi_{r(e_1, e_2)}^{(k+1)}$$

dim

è sufficiente definire

$$h' (e_1, e_2, \vec{x}, 0) = \varphi_{e_1}^{(k)}(\vec{x}) = \psi_0^{(k)}(e_1, \vec{x})$$

$$h' (e_1, e_2, \vec{x}, y+1) = \varphi_{e_2}^{(k+2)}(\vec{x}, y, f(\vec{x}, y)) = \psi_0^{(k+2)}(e_1, e_2, \vec{x}, y, h(e_2, \vec{x}, y))$$

h' , definita per ricorsione primitiva da funzioni calcolabili, è calcolabile

Se

il th. sorm. esiste $r: \mathbb{N}^2 \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$\varphi_{r(e_1, e_2)}(\vec{x}, y) = h'(e_1, e_2, \vec{x}, y)$$

come desiderato

Nota: Il fatto non è sorprendente: nella dimostrazione della chiusura di C rispetto all'operazioni di ric. primitiva, abbiamo mostrato come, dati

programmi F, G per f, g sia possibile costruire in modo effettivo un programma per h

la funzione è operata come segue:

- dati e_1, e_2

$$\text{trova } P_{e_1} = \gamma^{-1}(e_1), \quad P_{e_2} = \gamma^{-1}(e_2)$$

- applica la costruzione descritta nella dim., ottenendo un programma P per la funzione h

- restituisce $e = \gamma(P)$

↓
Intuitivo, ma difficilissima da scrivere formalmente: con Th. s.m.m. e funzione universale ce l'abbiamo gratis.

4) Effettività dell'inversa

Esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ totale calcolabile t.c.

$$\forall x \in \mathbb{N} \quad \text{se } \varphi_x \text{ è imiettiva} \Rightarrow \varphi_{K(x)} = (\varphi_x)^{-1}$$

dim

definiamo una funzione $g(x, y)$ che per x t.c. φ_x imiettiva soddisfi

$$g(x, y) = \varphi_x^{-1}(y) = \begin{cases} z & \text{se esiste } z \text{ t.c. } \varphi_x(z) = y \\ \uparrow & \text{altrimenti} \end{cases}$$

può essere

$$\begin{aligned} g(x, y) &= (\mu w. \exists s(x, (w)_1, y, (w)_2))_1 \\ &= (\mu w. |Xs(x, (w)_1, y, (w)_2) - 1|)_1 \end{aligned}$$

soddisfa la proprietà (vedi es. prec. sull'inverso) ed è calcolabile

¶

per il teorema s.m.m. esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ calco totale t.c.

$$\varphi_{K(x)}(y) = g(x, y) = (\varphi_x^{-1})(y) \quad \text{se } \varphi_x \text{ imiettiva}$$

• manipolazione di domini / codomini

(a) Esiste una funz. calcolabile totale $s: \mathbb{N}^2 \rightarrow \mathbb{N}$ t.c.

$$W_{s(x,y)} = W_x \cup W_y$$

dim

vogliamo che $\varphi_{s(x,y)}(z) \downarrow$ sse $\varphi_x(z) \downarrow$ oppure $\varphi_y(z) \downarrow$

\exists

definiamo una funzione con x, y come argomenti che sottiene la proprietà

$$g(x, y, z) = \begin{cases} 0 & z \in W_x \text{ oppure } z \in W_y \\ 1 & \text{altrimenti} \end{cases}$$

$$= \begin{cases} 0 & \varphi_x(z) \downarrow \text{ oppure } \varphi_y(z) \downarrow \\ 1 & \text{altrimenti} \end{cases}$$

$$= \Omega (\mu w. "H(x, z, w) \vee H(y, z, w)")$$

$$= \Omega (\mu w. |\chi_{H(x, z, w)} \vee H(y, z, w) - 1|)$$

{ mta basta un solo w perché $\exists w_1. P_1(w_1) \vee \exists w_2. P_2(w_2)$

$$\exists w. P_1(w) \vee P_2(w)$$

g calcolabile, dunque per il Teor. SMM esiste $s: \mathbb{N}^2 \rightarrow \mathbb{N}$ calcolabile totale

$$\varphi_{s(x,y)}(z) = g(x, y, z)$$

Dunque (prova del move)

$$z \in W_{s(x,y)} \Leftrightarrow \varphi_{s(x,y)}(z) = g(x, y, z) \downarrow$$

sse $z \in W_x$ oppure $z \in W_y$

sse $z \in W_x \cup W_y$

$$\rightsquigarrow W_{s(x,y)} = W_x \cup W_y$$

(b) Esiste $K: \mathbb{N}^2 \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$\forall x, y \quad E_{K(x,y)} = E_x \cup E_y$$

dim

Si vuole che i valori assunti da $\varphi_{S(x,y)}$ siano sia quelli di φ_x che di φ_y .

Idea: simulo φ_x sui pari e φ_y sui dispari

Al solito definisco una funzione dove x, y sono esponenti

$$g(x, y, z) = \begin{cases} \varphi_x(z/2) & \text{se } z \text{ pari} \\ \varphi_y(\frac{z-1}{2}) & \text{se } z \text{ dispari} \end{cases}$$

calcolabile in quanto

$$= \psi_0(x, qt(z, z)) \cdot \overline{\text{sg}}(\text{rm}(z, z)) + \psi_0(y, qt(z, z)) \cdot \text{sg}(\text{rm}(z, z))$$

No! errore φ_x potrebbe essere indef. quando scelgo φ_y così che la funz. risulta indefinita

Invece

$$\begin{aligned} g(x, y, z) &= (\mu w. ((s(x, qt(z, z), (w)_1, (w)_2) \wedge z \text{ pari}) \vee \\ &\quad | \quad s(y, qt(z, z), (w)_1, (w)_2) \wedge z \text{ dispari}))_1 \\ &= (\mu w. |\max| \chi_s(x, qt(z, z), (w)_1, (w)_2) \cdot \overline{\text{sg}}(\text{rm}(z, z)), \\ &\quad | \quad \chi_s(y, qt(z, z), (w)_1, (w)_2) \cdot \text{sg}(\text{rm}(z, z)) \}|^{-1})_1 \end{aligned}$$

teorema smm., esiste $K: \mathbb{N}^2 \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$\varphi_{K(x,y)}(z) = g(x, y, z)$$

Dunque

$$N \in E_{K(x,y)} \iff \exists z \quad \varphi_{S(x,y)}(z) = g(x, y, z) = N$$

$$\iff \exists z. \quad z \text{ pari e } \varphi_x(z/2) = N$$

$$\quad z \text{ dispari e } \varphi_y(z-1/2) = N$$

$$\iff \exists z. \quad \varphi_x(z) = N \vee \varphi_y(z) = N \iff \omega \in E_x \cup E_y$$

c) Esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$E_{K(x)} = W_x$$

dim

vogliamo che $y \in W_x$ sse $y \in E_{K(x)}$

$$\varphi_x(y) \downarrow$$

$$\exists z \quad \varphi_{K(x)}(z) = y$$

dato che la funzione lo costruiamo noi, lo z può essere proprio y

difiniamo

$$g(x, y) = \begin{cases} y & y \in W_x \\ \uparrow & \text{altrimenti} \end{cases}$$

$$= \perp(\psi_0(x, y)) \cdot y$$

$$\text{oppure } \psi_0(x, y) - \psi_0(x, y) + y$$

computabile

→ Teor. 5mm esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ comp. totale t.c.

$$\varphi_{K(x)}(y) = g(x, y)$$

si ha quindi

$$y \in E_{K(x)} \iff \varphi_{K(x)}(y) = y \iff g(x, y) = y \iff y \in W_x$$

\uparrow

$\varphi_{K(x)}$ è l'idefinita dove
è definita!

c') Esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ calc. totale t.c.

$$\forall x \quad \text{se } 0 \in W_x \text{ allora}$$

$$\varphi_{K(x)} \text{ totale e } E_{K(x)} = W_x$$

d) Data $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile, esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ calcol totale

t.c.

$$\forall x \quad W_{K(x)} = f^{-1}(W_x)$$

dim

vogliamo che

$$y \in W_{K(x)} \text{ sse } f(y) \downarrow \text{ e } f(y) \in W_x$$

...
...
...

$$\varphi_{K(x)}(y) \downarrow$$

$$\varphi_x(f(y)) \downarrow$$

quindi si definisce

$$g(x, y) = \varphi_x(f(y)) = \varphi_0(x, f(y))$$

calcolabile, dunque vi è, per il Teor. s.m.m. $K: \mathbb{N} \rightarrow \mathbb{N}$ calcol totale tale che

$$\varphi_{K(x)}(y) = g(x, y)$$

Quindi

$$y \in W_{K(x)} \text{ sse } \varphi_{K(x)}(y) = g(x, y) = \varphi_x(f(y)) \downarrow$$

$$\text{sse } f(y) \downarrow \text{ e } f(y) \in W_x$$

$$\text{sse } y \in f^{-1}(W_x)$$

□

* Operazioni sui predicatori

Esiste $K: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale t.c.

se $\varphi_x = \chi_Q$ è la funzione caratteristica di un predicato decidibile Q

$$\text{allora } \varphi_{K(x)} = \chi_{\neg Q}$$

dim

definiamo

$$g(x, y) = 1 - \varphi_x(y) = 1 - \varphi_0(x, y)$$

calcolabile!

dunque per il th. s.m.m. si ha κ calcolabile totale t.c.

$$g(x,y) = \varphi_k(x)$$

dunque se $\varphi_x = \pi_Q$

$$\varphi_k(x)(y) = 1 \iff \varphi_x(y) = 0 \iff \pi_Q(y) = 0$$

$$g(x,y) = 1 - \varphi_x(y)$$

2

$$\varphi_k(x) = \pi_{\neg Q}$$

Finora abbiamo visto varie funzioni calcolabili, problemi decidibili, ma solo in pochi casi abbiamo fornito esempi dell'ampio classe di funzioni non calcolabili e predicati non decidibili.

Obiettivo: iniziare uno studio matematico di

- classi di problemi imdecidibili
- tecniche per dimostrare l'imdecidibilità

in modo da evidenziare i limiti delle capacità dei calcolatori, e dare una struttura alle varie classi di problemi.

Σ

vedremo che la regola è: "quasi tutto quello che è interessante non è decidibile".

Ci focalizzeremo principalmente su insiemi di numeri $X \subseteq \mathbb{N}$

interrogandoci sul problema: "quanto è difficile stabilire se $x \in X$ "?

Questo ci porterà all'individuazione di

- insiemi ricorsivi "facili"
- insiemi ricorsivamente enumerabili "medi"

anche fra i problemi "difficili", imdecidibili si possono stabilire dei gradi di difficoltà

insiemi creativi, produttivi e semplici

Nota: Tutto questo ha un'ovvia formulazione in termini di predicati (un predicato umano non è altro che un sottoinsieme di \mathbb{N})

predicati decidibili

semi decidibili

moi ci concentriamo principalmente sugli insiemi!

* insiemi Ricorsivi

Def: Un insieme $A \subseteq \mathbb{N}$ si dice ricorsivo se la sua funzione caratteristica è calcolabile.

$$\chi_A : \mathbb{N} \rightarrow \mathbb{N}$$

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

è calcolabile

(ovvero se il problema/predicato " $x \in A$ " è decidibile)

Note: • se $\chi_A \in PR$, diciamo che A è primitivo ricorsivo.

• la nozione si estende in modo ovvio a sottoinsiemi di \mathbb{N}^k , tendenzialmente ci limiteremo a sottoinsiemi di \mathbb{N} , eventualm. codificando \mathbb{N}^k in \mathbb{N} !

Esempi

* sono ricorsivi

(a) \mathbb{N}

$$\chi_{\mathbb{N}}(x) = 1 \quad \forall x \quad \text{calcolabile!}$$

(b) numeri primi

$$P(x) = \begin{cases} 1 & \text{se } x \text{ primo} \\ 0 & \text{altrimenti} \end{cases} \quad \text{calcolabile}$$

(c) ogni insieme finito (esercizio)

Dato $A \subseteq_{fin} \mathbb{N}$ $A = \{x_1, \dots, x_m\}$

$$\chi_A(x) = \bigwedge_{i=1}^m (x - x_i)$$

* non sono ricorsivi

(a) $K = \{x \mid x \in \mathbb{N}_x\}$

abbiamo visto che

$$\chi_K(x) = \begin{cases} 1 & x \in \mathbb{N}_x \\ 0 & x \notin \mathbb{N}_x \end{cases}$$

non è calcolabile

(b) $\{x \mid \varphi_x \text{ totale}\}$

OSS.: Proprietà di chiusura degli insiemi ricorsivi

Se $A, B \subseteq \mathbb{N}$ ricorsivi allora sono ricorsivi

- 1) $\bar{A} = \mathbb{N} \setminus A$
- 2) $A \cap B$
- 3) $A \cup B$

dimm

dalle analoghe proprietà di chiusura dei predici decidibili

* Processo di Riduzione

E' un meccanismo semplice ma fondamentale nello studio di pb. di decidibilità.

Int.: tenta di formalizzare l'intuizione secondo cui un problema A è più semplice di un altro B

- se A è più facile di B e A imdecidibile $\Rightarrow B$ imdecidib.
- se A è .. " .. B e B decidibile $\Rightarrow A$ decidibile

L'idea è che A è più facile di B se una istanza di A può essere trasformata in modo facile in una istanza di B .

Σ

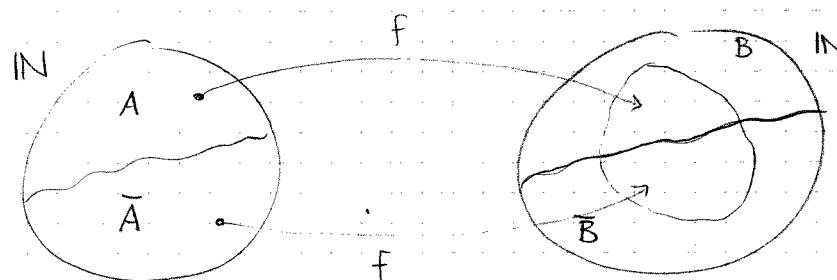
formalizzato dalla nozione di funzione di riduzione.

Def: Siamo $A, B \subseteq \mathbb{N}$ si dice che il problema $x \in A$ è riducibile al problema $x \in B$ (o semplicemente A è riducibile a B) e si scrive $A \leq_m B$

se esiste $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$x \in A \quad \text{sse} \quad f(x) \in B$$

(f si dice funzione di riduzione)



Proposizione: Siamo $A, B \subseteq \mathbb{N}$ t.c. $A \leq_m B$ allora

- 1) se B ricordivo $\Rightarrow A$ ricordivo
- 2) se A non ricordivo $\Rightarrow B$ non ricordivo

dim

basta osservare che

$$\chi_A = \chi_B \circ f$$

Esempio: Sappiamo che $K = \{x \mid x \in W_x\}$ non è ricordivo.

Vediamo come la non ricordatività di altri insiemi può essere dimostrata per riduzione a questo.

$$(1) K \leq \{\bar{x} \mid \varphi_{\bar{x}} \text{ totale}\}$$

A

dim

si deve dimostrare che esiste $f: \mathbb{N} \rightarrow \mathbb{N}$ computabile totale t.c.

$$x \in K \iff f(x) \in A$$

ovvero

$$x \in W_x \iff \varphi_{f(x)} \text{ totale}$$

si usa il teorema smm definiamo

$$g(x,y) = \begin{cases} 1 & x \in W_x \\ \uparrow & \text{altrimenti} \end{cases}$$

La funzione g è calcolabile; infatti

$$g(x,y) = \mathbf{1}(\varphi_x(x)) = \mathbf{1}(\psi_0(x,x))$$

dunque per il teorema smm esiste $s: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$\varphi_{s(x)}(y) = g(x,y)$$

ed s è proprio la funzione cercata; infatti

- $x \in K \Rightarrow x \in W_x \Rightarrow \forall y \varphi_{s(x)}(y) = g(x,y) = 1 \Rightarrow \varphi_{s(x)} \text{ totale} \Rightarrow s(x) \in A$
- $x \notin K \Rightarrow x \notin W_x \Rightarrow \forall y \varphi_{s(x)}(y) = g(x,y) \uparrow \Rightarrow \varphi_{s(x)} \text{ non totale} \Rightarrow s(x) \notin A$

(2) Consideriamo l'insieme

$$\{(x, y) \mid y \in W_x\}$$

(corrispondente al problema "y \in W_x")

per vederlo come sottoinsieme di \mathbb{N} occorre codificare le coppie come numeri.

per vedere come sottoinsieme di \mathbb{N} occorre codificare le coppie come numeri.

$$B = \{m \mid \pi_2(m) \in W_{\pi_1(m)}\}$$

$$(a) \underline{K \leq_m B}$$

(intuitivo: $x \in W_x$ più facile di $y \in W_x$!)

si deve trovare una funzione $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$m \in K \iff f(m) \in B$$

è sufficiente prendere

$$f(m) = \pi(m, m)$$

si ha allora

$$m \in K \Rightarrow m \in W_m \Rightarrow \pi_2(f(m)) \in W_{\pi_1(f(m))} \Rightarrow f(m) \in B$$

\Leftarrow

\Leftarrow

\Leftarrow

3

dato che K non ricorda $\rightsquigarrow B$ non ricorda

$$(b) B \leq_m K$$

(quindi sono ugualmente difficili...)

mt, dimenticando la codifica, dati $x, y \in \mathbb{N}$, devo costruire una funz. che termina sul suo indice $\iff y \in W_x$

definisco

$$g(x, y, z) = \begin{cases} 1 & y \in W_x \\ \uparrow & \text{altrimenti} \end{cases}$$

è calcolabile. Infatti

$$g(x, y, z) = \mathbb{1}(\varphi_x(y)) = \mathbb{1}(\varphi_0(x, y))$$

dunque per il th. s.m.m. $\exists s: \mathbb{N}^2 \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$g(x, y, z) = \varphi_{s(x, y)}(z)$$

La funzione di riduzione sarà dunque

$$f(m) = s(\pi_1(m), \pi_2(m))$$

infatti

- $m \in B \Rightarrow \pi_2(m) \in W_{\pi_1(m)} \Rightarrow \varphi_{f(m)}(z) = g(\pi_1(m), \pi_2(m), z) = 1 \quad \forall z$
 \Rightarrow in particolare $f(m) \in W_{f(m)}$ $\Rightarrow f(m) \in K$
- $m \notin B \Rightarrow \pi_2(m) \notin W_{\pi_1(m)} \Rightarrow \varphi_{f(m)}(z) = g(\pi_1(m), \pi_2(m), z) \uparrow \forall z$
 \Rightarrow in particolare $f(m) \notin W_{f(m)} = \emptyset \Rightarrow f(m) \notin K$

(3) consideriamo il problema

$$P(x, y) = " \varphi_x = \varphi_y "$$

Ancoara, mediante codifica possiamo ottenere un corrispondente insieme

$$C = \{m \mid \varphi_{\pi_1(m)} = \varphi_{\pi_2(m)}\}$$

dimostriamo che

$$K \leq_m C$$

Definiamo

$$g(x, y) = \begin{cases} 1 & x \in W_x \\ \uparrow & \text{altrimenti} \end{cases} = \mathbb{1}(\varphi_x(x)) = \mathbb{1}(\varphi_0(x, x))$$

essendo calcolabile, per il th. somm. inoltre $s : \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale

t.c.

$$\varphi_{s(x)}(y) = g(x, y)$$

ovvero $\varphi_{s(x)}$ è $\begin{cases} \text{la costante 1} & \text{se } x \in W_x \\ \text{la funz. sempre indef.} & \text{altrimenti} \end{cases}$

\therefore

farciamo in modo di confrontare $\varphi_{s(x)}$ con la costante 1

Sappiamo che $\mathbb{1}$ è calcolabile; sia e_1 un indice t.c.

$$\varphi_{e_1} = \mathbb{1}$$

Allora

$$f(m) = \pi(s(m), e_1)$$

si ha

$$\bullet m \in K \Rightarrow m \in W_m \Rightarrow \varphi_{s(m)} = 1 = \varphi_{e_1} \Rightarrow \pi(s(m), e_1) \in C \\ f(m)$$

$$\bullet m \notin K \Rightarrow m \notin W_m \Rightarrow \varphi_{s(m)} = \emptyset \neq \varphi_{e_1} \Rightarrow \pi(s(m), e_1) \notin C$$

$$(4) \{x : \varphi_x = 1\} \leq_m C$$

infatti, detto e_1 un indice per la funzione $\mathbb{1}$, la funzione di riduzione può essere

$$f(x) = \pi(x, e_1)$$

(5) Problema dell'input

Dimostrare che $\forall m \in \mathbb{N}$

$$A_m = \{x : \varphi_x(m) \downarrow\} \quad m \text{m ricorsivo}$$

\uparrow determinazione su di
uno specifico input

si dimostra

$$K \leq_m A_m$$

dobbiamo definire f t.c.

$$x \in K \text{ sse } f(x) \in A_m$$

$$x \in W_x \quad \varphi_{f(x)}(m) \downarrow$$

Difiniamo

$$g(x, y) = \begin{cases} 1 & x \in W_x \\ \uparrow & \text{altrimenti} \end{cases}$$

$$= \mathbb{1}(\varphi_y(x, x))$$

calcolabile \Rightarrow per il th. smm esiste $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile t.c.

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

dunque

$$\bullet x \in K \Rightarrow \varphi_{f(x)}(y) = g(x, y) = 1 \quad \forall y \Rightarrow \varphi_{f(x)}(m) = 1 \downarrow \Rightarrow f(x) \in A_m$$

$$\bullet x \notin K \Rightarrow \varphi_{f(x)}(y) = g(x, y) \uparrow \quad \forall y \Rightarrow \varphi_{f(x)}(m) \uparrow \Rightarrow f(x) \notin A_m$$

(6) Problema dell'output

Dato $m \in \mathbb{N}$... $B_m = \{x \mid m \in E_x\}$... mom è ricorsivo

dim

dimostriamo che

$$K \leq_m B_m$$

difiniamo

$$g(x, y) = \begin{cases} m & x \in W_x \\ \uparrow & \text{altrimenti} \\ 1 & \end{cases} = m \cdot \mathbf{1}(\psi_g(x, x))$$

calcolabile \Rightarrow il teorema s.m.m ci fornisce $s: \mathbb{N} \rightarrow \mathbb{N}$ t.c.

$$\forall y \quad g(x, y) = \varphi_{s(x)}(y)$$

e vale

$$x \in K \Rightarrow x \in W_x \Rightarrow \varphi_{s(x)}(y) = m \quad \forall y \Rightarrow m \in E_{s(x)} = \{m\} \Rightarrow s(x) \in B_m$$

$$x \notin K \Rightarrow x \notin W_x \Rightarrow \varphi_{s(x)}(y) \uparrow \quad \forall y \Rightarrow m \notin E_{s(x)} = \emptyset \Rightarrow s(x) \notin B_m$$

OSSERVAZIONE: Siamo $A, B \subseteq \mathbb{N}$ con $A \leq_m B$ tramite una funzione

di riduzione $f: \mathbb{N} \rightarrow \mathbb{N}$ (comp. totale) imiettiva.

Dato che f' calcolabile, si potrebbe pensare che anche $B \leq_m A$.

In realtà f' mom è totale dunque riduce ad A un "sottoproblema" di B .

che, in principio, mom ha relazioni di completezza con B .

Es:

$P =$ insieme dei numeri pari

$$K_3 = \{x \mid (\text{rm}(3, x) = 0 \text{ e } x/3 \in K) \vee (\text{rm}(3, x) = 1\}$$

Allora

$$\bullet \quad K \leq K_3$$

con funzione di riduzione $f(x) = 3 \cdot x$

$$x \in K \iff f(x) = 3 \cdot x \in K_3$$

- $P \leq K_3$

e la funzione di riduzione può essere la seguente

$$f(x) = \begin{cases} 3x+1 & x \text{ pari} \\ 3x+2 & x \text{ dispari} \end{cases}$$

che è imiettiva

- tuttavia $K_3 \leq P$

altrimenti, enendo P ricorsivo, K_3 sarebbe ricorsivo
 $\sim K$ ricorsivo.

Si noti che vale invece

- * Siamo $A, B \subseteq \mathbb{N}$ e sia $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale bigettiva
 t.c. $x \in A \iff f(x) \in B$

allora $A \leq B$ e $B \leq A$

Questo giustifica l'uso di codifiche nel trattamento della decidibilità di insiemi di simboli / predicatori matrici

(le funzioni di codifica $\mathbb{N}^K \rightarrow \mathbb{N}$ sono calcolabili biamivio che)

* Teorema di Rice

Il teorema di Rice fornisce un risultato di indecidibilità estremamente generale. Afferma sostanzialmente che qualsiasi proprietà delle funzioni calcolabili (tranne quelle banali) è decidibile.

Per formularlo in termini di insiemi di numeri naturali, abbiamo bisogno della definizione seguente:

Def (Insieme saturato)

Un sottoinsieme $A \subseteq \mathbb{N}$ si dice SATURATO se

$$\forall x, y \quad x \in A \wedge \varphi_x = \varphi_y \Rightarrow y \in A$$

(ovvero se $x \in A$ allora A contiene tutti gli indici per la funzione φ_x)

Detto ancora in altri termini, A è saturato se esprime una proprietà di tutti i programmi, indipendentemente dagli indici (programmi chi li calcolano).

$$A = \{x \mid P(\varphi_x)\}$$

indip. da x

o ancora, se esiste $d \in \mathbb{C}$ t.c.

$$A = \{x \mid \varphi_x \in d\}$$

Esempi

- $\{x \mid \varphi_x \text{ totale}\}$

- $\{x \mid \varphi_x = \perp\}$

sono saturati

- $K = \{x \mid x \in Wx\}$

non saturato, ma non è ovvio

(si potrebbe dimostrare che vi è un indice $e \in \mathbb{N}$ t.c.

$$\varphi_e = \{(e, 0)\} \quad \text{m. } e \in K$$

così che, tenendoci infiniti indici per φ_e , dato un qualsiasi $e' \neq e$

t.c. $\varphi_{e'} = \varphi_e \quad \text{m. } e' \notin K$

conseguenza di risultati successivi).

Vediamo quindi il teorema di "Rice", che dice che non c'è modo di decidere una proprietà non banale di insiemi di funzioni.

Teorema (Hemry Gordon Rice, 1953) (Syracuse, NY - dottorato)

Sia $A \subseteq \mathbb{N}$ saturo, $A \neq \emptyset$, $N \Rightarrow A$ non è ricorsivo.

dim

sia $\emptyset \neq A \subseteq \mathbb{N}$ A saturo.

dimostriamo che

$$K \leq_m A$$

Consideriamo la funzione sempre indefinita, che sappiamo essere calcolabile, e sia e_0 un suo indice

$$\varphi_{e_0}(x) \uparrow \quad \forall x$$

si distinguono due possibili casi, a seconda che lo sia o meno in A

1) lo $\notin A$

Consideriamo $e_1 \in A$ (esiste certamente dato che $A \neq \emptyset$)

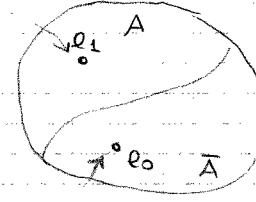
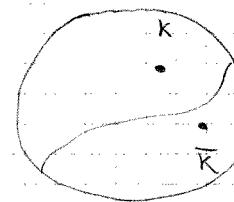
Idea: costruire una funz.

parametrica in x che

coincida con

$$-\varphi_{e_1} \text{ se } x \in K$$

$$-\varphi_{e_0} \text{ se } x \notin K$$



Ma poi con il th. s.m.m. si trova la funzione di riduzione

Basta definire

$$g(x,y) = \begin{cases} \varphi_{e_1}(x) & x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

La funzione g è calcolabile, infatti

$$g(x,y) = \varphi_{e_1}(x) : \mathbf{1}(\varphi_x(x)) = \varphi_{e_1}(x) : \mathbf{1}(\varphi_0(x,x))$$

dunque, per il teorema s.m.m. esiste $s: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$g(x,y) = \varphi_{s(x)}(y)$$

e s riduce K ad A infatti

$$\begin{aligned} \bullet x \in K &\Rightarrow x \in W_x \Rightarrow \varphi_{s(x)}(y) = g(x,y) = \varphi_{e_1}(x) \Rightarrow \varphi_{s(x)} = \varphi_{e_1} \text{ e } e_1 \in A \\ &\Rightarrow \text{entendo } A \text{ saturo } s(x) \in A \end{aligned}$$

$$\bullet x \notin K \Rightarrow x \notin W_x \Rightarrow \varphi_{s(x)}(y) = g(x,y) \uparrow \forall y \Rightarrow \varphi_{s(x)} = \varphi_{e_0} \text{ e } e_0 \notin A$$

pertanto $K \leq_m A$ ed entrodo K non riconoscibile $\Rightarrow A$ non riconoscibile

(2) $e_0 \notin A$

In questo caso, basta osservare che $e_0 \in \bar{A}$ e $\emptyset \neq \bar{A} \subseteq \mathbb{N}$

\Rightarrow per il punto precedente \bar{A} non riconoscibile

$\Rightarrow A$ non riconoscibile \square

Esempio: "Ex è infinito" non è decidibile

ovvero

$$A = \{x \mid \text{Ex infinito}\} \quad \text{non riconoscibile}$$

dim.

basta osservare che

- A saturo
- $A \neq \emptyset$

es: se e_I è un indice per la funzione identità $\varphi_{e_I} = \lambda x.x$

allora $E_{e_I} = \mathbb{N} \Rightarrow e_I \in A$

- $A \neq \mathbb{N}$

es: se e_0 è un indice per la funzione sempre indefinita, allora

$E_{e_0} = \emptyset \text{ m. } e_0 \notin A$

e si conclude per il teorema di Rice.

* Problema dell'output

$$A_m = \{x \mid m \in E_x\} \quad \text{non riconoscibile}$$

Infatti, si può usare il teorema di Rice dato che

- A_m saturo
- $A_m \neq \emptyset$ p.es. se e_m è un indice per $\lambda x.m$ $\Rightarrow e_m \in A_m$
- $A_m \neq \mathbb{N}$ se $e_k \in A_m$ $\exists x \in \mathbb{N} \quad \lambda x.k, x \neq m \Rightarrow e_k \notin A_m$

Non si deve pensare che il teorema di Rice non applichi ad ogni problema di decidibilità ... es...

28 Esercizio

Dimostrare che $A = \{x \mid x \in W_x \cap E_x\}$ non è ricorsivo

dim.

Si noti che A non è sottunito (o meglio non è ovvio se P_0 è a meno...)

dimostriamo che $K \leq_m A$

difiniamo

$$g(x, y) = \begin{cases} y & x \in W_x \\ \uparrow & \text{altrimenti} \end{cases}$$

$$= y * \mathbb{1}(p_x(x)) = y * \mathbb{1}(\varphi_0(x, x))$$

è computabile, dunque per il th. s.m.m. esiste $s: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile t.c.

$$g(x, y) = \varphi_{s(x)}(y)$$

ed s può essere la funzione di riduzione; infatti

$$\bullet x \in K \Rightarrow \varphi_{s(x)}(y) = g(x, y) = y \quad \forall y \Rightarrow \varphi_{s(x)}(s(x)) = s(x)$$

$$\Rightarrow s(x) \in W_{s(x)} \cap E_{s(x)} \Rightarrow s(x) \in A$$

$$\bullet x \notin K \Rightarrow \varphi_{s(x)}(y) = g(x, y) \uparrow \quad \forall y \Rightarrow \varphi_{s(x)}(s(x)) \uparrow$$

$$\Rightarrow s(x) \notin W_{s(x)} \Rightarrow s(x) \notin W_{s(x)} \cap E_{s(x)} \Rightarrow s(x) \notin A$$

□

INSIEMI RICORSIVAMENTE ENUMERABILI

Si tratta di una classe più ampia, che comprende gli insiemi ricorsivi: intuitivamente un insieme A è ricorsivamente enumerabile se alla dom. " $x \in A?$ "

occorre saper rispondere sì, quando questo è vero, ma se la risposta è negativa si può anche non dare alcuna risposta.

Def: Un insieme $A \subseteq \mathbb{N}$ si dice ricorsivamente enumerabile (r.e.) se la sua funzione semi-caratteristica

$$\text{SC}_A(x) = \begin{cases} 1 & x \in A \\ \uparrow & \text{altrimenti} \end{cases} \quad \text{è calcolabile.}$$

Analogamente, un predicato $P(x)$ si dice semi-decidibile se l'insieme $\{x \mid P(x)\}$ è r.e.

Le due nozioni ammettono una ovvia estensione al caso di

- sottinsiemi di \mathbb{N}^k
- prediciati k -ari

tramite codifiche o per definizione diretta, es. $P(x_1, \dots, x_k)$ semi-dec.

se $\text{SC}_P(x_1, \dots, x_k) = \begin{cases} 1 & \text{se } P(x_1, \dots, x_k) \\ \uparrow & \text{altrimenti} \end{cases} \quad \text{è calcolabile.}$

Dunque le nozioni di insieme r.e. / predicato semi-decidibile appaiono più deboli rispetto a insieme ricorsivo / predicato decidibile.

Questo è formalizzato nel seguente risultato:

Teorema: Sia $A \subseteq \mathbb{N}$

A è ricorsivo sse A e \bar{A} sono r.e.

dim

(\Rightarrow) Sia $A \subseteq \mathbb{N}$ ricorsivo, dunque la funzione caratteristica

$$\chi_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \quad \text{è calcolabile.}$$

Perb.mto

(1) A è r.e., infatti la funzione semi-caratteristica

$$\text{SC}_A(x) = \begin{cases} 1 & x \in A \\ \uparrow & \text{altrimenti} \end{cases}$$

si può esprimere come

$$SC_A(x) = \mathbb{1}(\mu z. (X_A(x) - z))$$

(2) \bar{A} è t.c.

infatti se A ricorsivo $\rightsquigarrow \bar{A}$ ricorsivo \rightsquigarrow per il punto (1) \bar{A} re.

(\Leftarrow) siamo A, \bar{A} re., quindi SC_A e $SC_{\bar{A}}$ sono calcolabili e pertanto anche

$1 - SC_{\bar{A}}$ d.f. da

$$1 - SC_{\bar{A}}(x) = 1 - SC_{\bar{A}}(x) = \begin{cases} 0 & x \in \bar{A} \\ 1 & \text{altrimenti} \end{cases}$$

è calcolabile.

dunque v. sono $e_0, e_1 \in \text{IN}$ t.c.

$$\varphi_{e_0} = SC_A \quad \text{e} \quad \varphi_{e_1} = 1 - SC_{\bar{A}}$$

Idea: si alternano i passi delle due macchine su x , fino a che una mormo termina; si osservi che una termima certamente dato che $x \in A$ oppure $x \in \bar{A}$!

$$\begin{aligned} X_A(x) &= (\mu w. (s(e_0, x, (w)_1, (w)_2) \vee s(e_1, x, (w)_1, (w)_2)))_1 \\ &= (\mu w. (\mathbb{1}X_{s(e_0, x, (w)_1, (w)_2)} \vee s(e_1, x, (w)_1, (w)_2) - 1))_1 \end{aligned}$$

calcolabile!

$\rightsquigarrow A$ è ricorsivo \square

OSSERVAZIONE: l'insieme $K = \{x \mid x \in W_x\}$ è t.c.; infatti

$$SC_K(x) = \begin{cases} 1 & x \in K \\ 0 & \text{altrimenti} \end{cases} = \mathbb{1}(\varphi_x(x)) = \mathbb{1}(\psi_0(x))$$

è calcolabile

Sappiamo che K mormo è ricorsivo; dunque per il teorema appena dimostrato

$$K = \{x \mid x \notin W_x\} \quad \text{mormo} \quad \text{t.c.}$$

Intuitiv: stabilire se $\varphi_x(x) \downarrow$ è più facile che mormo stabilire se $\varphi_x(x) \uparrow$

Continuiamo con alcuni risultati riguardanti i predici semi-decidibili

* Teorema di struttura dei predici semi-decidibili

Un predico $P(\bar{x}) \in \text{IN}^k$ è semi-decidibile
sse

esiste un predico $Q(t, \bar{x}) \in \text{IN}^{k+1}$ t.c. $P(\bar{x}) \equiv \exists t. Q(t, \bar{x})$
decidibile

dim

(\Rightarrow) sia $P(\bar{x})$ semi-decidibile;

dunque la funzione semi-caratteristica s_p è calcolabile
esiste $e \in \text{IN}$ t.c. $s_p = q_e^{(k)}$

per def. di funzione semi-caratteristica

$P(\bar{x})$ sse il programma p_e su \bar{x} termina (e produce 1)

ovvero $P(\bar{x}) \equiv \exists t. H(e, \bar{x}, t)$

Dunque, ponendo $Q(t, \bar{x}) = H(e, \bar{x}, t)$, Q è decidibile e
come desiderato

$$P(\bar{x}) \equiv \exists t. Q(t, \bar{x})$$

(\Leftarrow) Sia $P(\bar{x}) \equiv \exists t. Q(t, \bar{x})$ con $Q(t, \bar{x})$ decidibile

Si ha quindi

$$\begin{aligned} s_p(\bar{x}) &= \mathbb{1}(\mu t. "Q(t, \bar{x})") \\ &= \mathbb{1}(\mu t. |\chi_Q(t, \bar{x}) - 1|) \end{aligned}$$

Dunque s_p è calcolabile (in quanto χ_Q lo è) e perciò
 $P(\bar{x})$ è semi-decidibile.

□

Da questo si deduce immediatamente che lo classe dei predici semi-dec.
è chiusa per quantificazione esistenziale, ovvero

Teorema di proiezione: Sia $P(x, \bar{y})$ un predico semi-decidibile.

Allora

$$P(\bar{y}) = \exists x. P(x, \bar{y})$$

è semi-decidibile.

dimm

Sia $P(x, \vec{y})$ semi-decidibile; dunque, per il teorema di struttura esiste $Q(t, x, \vec{y})$ decidibile t.c.

$$P(\vec{x}, \vec{y}) \equiv \exists t. Q(t, x, \vec{y})$$

quindi se $P'(\vec{y}) = \exists x. P(x, \vec{y})$ vale

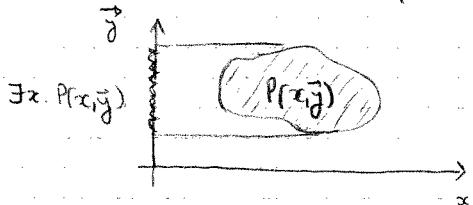
$$P'(\vec{y}) \equiv \exists x. \exists t. Q(t, x, \vec{y})$$

$$\equiv \exists \omega. Q((\omega)_1, (\omega)_2, \vec{y})$$

chiaramente questo è un predicato decidibile

→ ancora per il teorema di struttura $P'(\vec{y})$ semidecidibile. □

OSSERVAZIONE: Perché "proiezione"?



Teorema (Proprietà di Chiusura)

Siamo $P_1(\vec{x})$ e $P_2(\vec{x})$ prediciati semi-decidibili. Allora anche i prediciati

$$(1) P_1(\vec{x}) \wedge P_2(\vec{x})$$

$$(2) P_1(\vec{x}) \vee P_2(\vec{x})$$

sono semi-decidibili.

dimm

Siamo $P_1(\vec{x}), P_2(\vec{x})$ semi-decidibili. Dunque per il teorema di struttura vi sono due prediciati decidiibili

$$Q_1(t, \vec{x}), Q_2(t, \vec{x})$$

tali che

$$P_1(\vec{x}) \equiv \exists t. Q_1(t, \vec{x})$$

$$P_2(\vec{x}) \equiv \exists t. Q_2(t, \vec{x})$$

Quindi

$$(1) P_1(\vec{x}) \wedge P_2(\vec{x}) \equiv \exists t. Q_1(t, \vec{x}) \wedge \exists t. Q_2(t, \vec{x})$$

$$\equiv \exists \omega. (Q_1((\omega)_1, \vec{x}) \wedge Q_2((\omega)_2, \vec{x}))$$

decidibile (Q_1, Q_2 decidi + composizione)

quindi per il teorema di struttura $P_1(\bar{x}) \wedge P_2(\bar{x})$ semi-decidibile.

(2) Analogo (\Leftarrow più semplice)

$$\begin{aligned} P_1(\bar{x}) \vee P_2(\bar{x}) &\equiv \exists t. Q_1(t, \bar{x}) \vee \exists t. Q_2(t, \bar{x}) \\ &\equiv \exists t. (\underbrace{Q_1(t, \bar{x}) \vee Q_2(t, \bar{x})}_{\text{decidibile!}}) \end{aligned}$$

OSSERVAZIONE: Dunque l'insieme dei predicati semi-decidibili è chiuso rispetto a \wedge, \vee, \exists

* mom è chiuso, invece, rispetto a \forall

ovvero se $P(x, y)$ semi-decidibile (o anche decidibile)

$$\Rightarrow \forall x. P(x, y) \text{ semi-decidibile}$$

Es.: se $P(x, t) = \neg H(x, x, t)$ decidibile

3

$$\forall t. P(x, t) = x \in \bar{K} \quad \text{mom semi-dec.}$$

* mom è chiuso rispetto a \neg (ovviamente, altrimenti sarebbe chiuso rispetto a \forall)

Es.: $x \in K$ semi-decidibile

$$x \notin K \equiv x \in \bar{K} \quad \text{mom semi-decid.}$$

OSSERVAZIONE: I risultati precedenti hanno una ovvia traduzione in termini di proprietà degli insiemmi r.e.

(1) $A \subseteq \mathbb{N}$ è rcozivo se e solo se A, \bar{A} r.e.

(2) se $A \subseteq \mathbb{N}$ r.e. e $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile $\Rightarrow f(A)$ è r.e. (proiezione)

(3) se $A, B \subseteq \mathbb{N}$ r.e. $\Rightarrow A \cup B, A \cap B$ sono r.e.

ND

* Iinsiemi ce. e riducibilità

Valgono proprietà analoghe a quelle già viste per gli insiemi ricorsivi, ovvero

* Siamo $A, B \subseteq \mathbb{N}$ con $A \leq_m B$. Allora

(1) se B è r.e. $\Rightarrow A$ è r.e.

(2) se A non r.e. $\Rightarrow B$ non r.e.

dim

(1) sia $f: \mathbb{N} \rightarrow \mathbb{N}$ la funzione di riduzione (calcolabile, totale) t.c.
 $x \in A \iff f(x) \in B$

se B è r.e. allora S_B è calcolabile, dunque

$$S_A = S_B \circ f$$

composizione di funzioni calcolabili, è calcolabile

$\Rightarrow A$ r.e.

(2) contraddim. male

□

* Caratterizzazioni alternative degli insiemi r.e.

Vediamo ora due caratterizzazioni alternative degli insiemi r.e.

La prima motiva il nome "ricorsivamente enumerabile", dato che caratterizza gli insiemi r.e. come immagini di enumerazioni (effettive).

Teorema: Sia $A \subseteq \mathbb{N}$

A è r.e. $\iff A = \emptyset$ oppure A è l'immagine di una funz. calc. totale

(\exists)
gli elem. di A possono essere enumerati
in modo effettivo: $f(0), f(1), f(2), \dots$

(\Rightarrow) Sia $A \subseteq \mathbb{N}$ r.e.; se $A = \emptyset$ ok

Sia dunque $A \neq \emptyset$; poiché A è r.e. la funzione semi-corrett. S_A è calcolabile \Rightarrow esiste $\varphi \in \mathbb{N}$ t.c.

$$S_A = \varphi_e$$

sia $z_0 \in A$ fissato (viene automaticamente dato che $A \neq \emptyset$) e definiamo

$$f(z) = \begin{cases} (z)_1 & \text{se } \varphi_e((z)_1) \downarrow \text{ in } (z)_2 \text{ o meno poss.} \\ a_0 & \text{altrimenti} \end{cases}$$

$$= \begin{cases} (z)_1 & \text{se } H(e, (z)_1, (z)_2) \\ 0 & \text{altrimenti} \end{cases}$$

$$= (z)_1 \cdot \chi_H(e, (z)_1, (z)_2) + 0 \cdot \chi_{\neg H}(e, (z)_1, (z)_2)$$

Dunque f è calcolabile, totale e $\text{im}(f) = A$ infatti

$$\text{im}(f) \subseteq A \quad \forall z. \quad \text{se } f(z) = 0 \Rightarrow f(z) \in A$$

$$\begin{aligned} \text{se } f(z) = (z)_1 &\Rightarrow H(e, (z)_1, (z)_2) \\ &\Rightarrow \varphi_f((z)_1) \downarrow \Rightarrow (z)_1 \in A \\ &\text{SCA}((z)_1) \end{aligned}$$

$$\begin{aligned} A \subseteq \text{im}(f) &\quad \text{se } x \in A \Rightarrow \exists t. \quad H(e, x, t) \Rightarrow \\ &\Rightarrow \text{posto } z = 2^x 3^t \quad f(z) = (z)_1 = x \\ &\Rightarrow x \in \text{im}(f) \end{aligned}$$

(\Leftarrow) se $A = \emptyset \Rightarrow A$ è ricorsivo \sim r.e.

se $A = \text{im}(f)$ con $f: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale, allora
 $x \in A \iff x \in \text{im}(f) \iff \exists z. \quad f(z) = x$

e dunque, essendo f totale, possiamo scrivere la funzione semi-cotott.
di A come

$$\text{SCA}(x) = \mathbb{I}(\mu z. |f(z)-x|)$$

Dato che f è calcolabile, SCA è calcolabile $\Rightarrow A$ è r.e.

[NB. per dim che $A \vee B \Rightarrow C$ si dim $A \Rightarrow C$ e $B \Rightarrow C$
usando il fatto che

$$(A \vee B) \Rightarrow C = (A \Rightarrow C) \wedge (B \Rightarrow C)$$

Teorema (insiemi r.e. come dominio)

$A \subseteq \mathbb{N}$ è r.e. sse A è il dominio di una funzione calcolabile
dim.

(\Rightarrow) sia A r.e., allora pur dif. la funz. semi-cotott. di A

$$\text{SCA}(x) = \begin{cases} 1 & x \in A \\ 0 & \text{altrimenti} \end{cases} \quad \text{è calcolabile}$$

(\Leftarrow) Sia $A = \text{dom}(f)$, con f calcolabile.

Allora SCA può essere scritta come

$$\text{SCA}(x) = \mathbb{1}(f(x))$$

ossia $\text{SCA} = \mathbb{1} \circ f$, composizione di funzioni calcolabili \rightarrow calcolabile

$\rightsquigarrow A$ è re. □

TEOREMA di RICE e SHAPIRO

Sappiamo che, per il teorema di Rice, nessuna proprietà delle funzioni calcolabili (tranne quelle banali) è decidibile.

Il teorema di Rice-Shapiro ci dice che tali proprietà possono essere semi-decidibili solo quando dipendono da una parte finita della funzione (ovvero dal valore che la funzione assume su di un numero finito di valori).

Ci serve dunque la nozione di funzione finita...

Def: Una funzione finita è per noi una funzione $\theta : \mathbb{N} \rightarrow \mathbb{N}$ t.c. $\text{dom}(\theta)$ è finito
(dunque θ è finita come insieme...)

$$\exists \text{ del tipo } \theta(x) = \begin{cases} y_1 & \text{se } x = x_1 \\ y_k & \text{se } x = x_k \\ \uparrow & \text{altrimenti} \end{cases}$$

Date due funzioni $f, g : \mathbb{N} \rightarrow \mathbb{N}$ scriviamo $f \leq g$ per dire che l'inclusione vale in senso insiemistico

$\rightsquigarrow f \leq g$ sse $\forall x. f(x) \downarrow \Rightarrow g(x) \downarrow$ ed in questo caso $f(x) = g(x)$

Norman Shapiro, studente di Church, nella sua tesi di dottorato
grado di riducibilità

Teorema (Rice-Shapiro)

Sia $A \subseteq \mathcal{C}$ un insieme di funzioni calcolabili

Se l'insieme $\{x \mid \varphi_x \in A\}$ è r.e. allora

$\forall f. (f \in A \iff \exists \theta \text{ funzione finita, } \theta \leq f \text{ e } \theta \in A)$

dim

asserto un po' complicato... sia $A = \{x \mid \varphi_x \in A\}$

difiniti i predicati

$P = "A \text{ è r.e.}"$

$Q(f) = "f \in A"$

$R(f) = "\exists \theta. \theta \text{ finita, } \theta \leq f, \theta \in A"$

dobbiamo dimostrare che

$$P \Rightarrow (\forall f. Q(f) \Leftrightarrow R(f))$$

quello che faremo sarà dimostrare la contramimale, ovvero:

$$\neg (\forall f. Q(f) \leftrightarrow R(f)) \rightarrow \neg P$$

$$\exists f. \neg (Q(f) \leftrightarrow R(f)) \rightarrow \neg P$$

$$\exists f. (\neg Q(f) \wedge R(f) \vee Q(f) \wedge \neg R(f)) \rightarrow \neg P$$

$$(\exists f. \neg Q(f) \wedge R(f) \vee \exists f. Q(f) \wedge \neg R(f)) \rightarrow \neg P$$

$$(\exists f. \neg Q(f) \wedge R(f) \rightarrow \neg P) \wedge (\exists f. Q(f) \wedge \neg R(f) \rightarrow \neg P)$$

(1)

(2)

① $\exists f. f \notin A$ e $\exists \vartheta$ finita, $\vartheta \in f$ e $\vartheta \in A \Rightarrow A$ mom t.c.

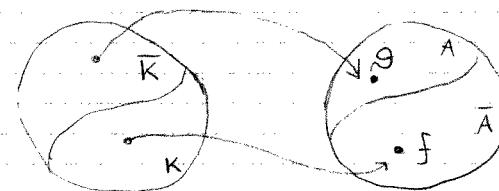
sia $f \notin A$ e sia ϑ finita, $\vartheta \in f$, $\vartheta \in A$

dimostriamo che A mom t.c. proviamo che

$$R \leq A$$

idea: definiamo una funzione "parametrica" in x , che coincide con

- ϑ quando $x \in \bar{K}$ ($x \notin K$)
- f quando $x \in K$



definiamo

$$g(x,y) = \begin{cases} f(y) & \text{se } x \in K \vee y \in \text{dom}(\vartheta) \\ \uparrow & \text{altrimenti} \end{cases}$$

si nota che il predicato $Q(x,y) = x \in K \vee y \in \text{dom}(\vartheta)$ è semi dec.

infatti

- $x \in K$ semi dec.
- $y \in \text{dom}(\vartheta)$ decidibile, in quanto $\text{dom}(\vartheta)$ finito

dunque la funzione semi-caratteristica s.a. è calcolabile, pertanto

$$g(x,y) = f(y) \cdot s_Q(x,y)$$

è calcolabile;

Per il teorema s.m. esiste una funzione calcolabile totale $S: \mathbb{N} \rightarrow \mathbb{N}$ t.c.

$$g(x,y) = \varphi_{S(x)}(y)$$

e s'può avere la funzione di riduzione; infatti

- $x \in \bar{K} \Rightarrow x \notin K \Rightarrow \varphi_{s(x)}(y) = g(x, y) = \emptyset(y) \quad \forall y$
 $\Rightarrow \varphi_{s(x)} = \emptyset \in \mathcal{A} \Rightarrow s(x) \in A$
- $x \notin \bar{K} \Rightarrow x \in K \Rightarrow \varphi_{s(x)}(y) = g(x, y) = f(y) \quad \forall y$
 $\Rightarrow \varphi_{s(x)} = f \notin \mathcal{A} \Rightarrow s(x) \notin A$

② Es. f.c.d. e $\forall \emptyset$ finita, $\emptyset \subseteq f$ $\emptyset \notin A \Rightarrow A$ non r.e.

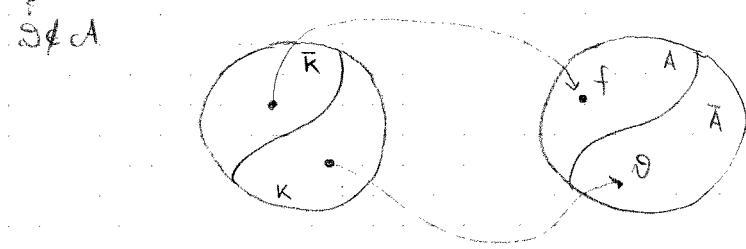
sia f.c.d. l.c. $\forall \emptyset$ finita, $\emptyset \subseteq f$ $\emptyset \notin A$;

dimostriamo che A non è r.e. provando che

$$\bar{K} \leq A$$

Idea: definiamo una funzione parametrica in x che è

- f quando $x \in \bar{K}$
- una funzione finita $\emptyset \subseteq f$ quando $x \notin \bar{K}$



definiamo

$$h(x, t) = \begin{cases} f(t) & \text{se } \neg H(x, x, t) \\ \uparrow & \text{altrimenti} \end{cases}$$

$f \text{ se } \varphi_x(x) \uparrow (x \in \bar{K})$
 $\emptyset \subseteq f \text{ altrimenti.}$

$$= f(t) \cdot \mathbb{1}(\mu y. \chi_{H(x, x, t)})$$

h è calcolabile, dunque per il th. s.m.m. esiste $s: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale l.c.

$$h(x, t) = \varphi_{s(x)}(t)$$

e la funzione s può avere la funzione di riduzione. Infatti

- $x \in \bar{K} \Rightarrow \varphi_x(x) \uparrow \Rightarrow \forall t. \neg H(x, x, t) \Rightarrow$
 $\Rightarrow \forall t. \varphi_{s(x)}(t) = h(x, t) = f(t)$
 $\Rightarrow \varphi_{s(x)} = f \in \mathcal{A} \Rightarrow s(x) \in A$

$\bullet \quad x \notin K \Rightarrow \varphi_x(x) \downarrow \Rightarrow \exists t_0 \text{ t.c. } \forall t \geq t_0 H(x, x, t) \wedge \forall t < t_0 \neg H(x, x, t)$
 $\Rightarrow \begin{cases} \forall t \geq t_0 \varphi_{S(x)}(t) = h(x, t) \uparrow \\ \forall t < t_0 \varphi_{S(x)}(t) = h(x, t) = f \end{cases}$
 $\Rightarrow \begin{cases} \text{dom } (\varphi_{S(x)}) \subseteq [0, t_0] \text{ finito.} \\ \varphi_{S(x)} \in f \end{cases}$
 $\Rightarrow \varphi_{S(x)} \notin A \Rightarrow S(x) \notin A \quad \square$

Esempio :

(1) $A = \{x \mid \varphi_x \text{ totale}\}$ mom i.e.

dim

chiaramente A è sottovolume e l'insieme di funzioni da cui deriva è

$$C = \{f \in C \mid f \text{ totale}\}$$

ovvero

$$A = \{x \mid \varphi_x \in C\}$$

Data una qualsiasi funzione $f \in C$ (quindi totale), ad es. $f = \text{id} = \lambda x.x$

chiaramente $\forall \theta \in f$ finita $\theta \notin A$, dato che ogni funz. finita è parziale
per il teor di Rice-Shapiro A mom i.e.

Alternativamente, poiché se \emptyset a dom. finito $\sim \emptyset \notin A$

se A è allora $f \in A$ solo se $\exists \theta \in f \theta$ finita $\theta \in A$

ovviamente per ovvio A è comunque $A = \emptyset$

(2) $\bar{A} = \{x \mid \varphi_x \text{ mom totale}\}$ mom i.e.

sia $\bar{A} = \{f \in C \mid f \text{ mom totale}\}$

osserviamo che ogni θ finita è in \bar{A} , ma innumere entensione totale
di tali θ può appartenere ad \bar{A} ! si conclude per Rice-Shapiro

più precisamente, si può osservare che

$$\begin{aligned} - \emptyset \text{ (finita)} &\in \bar{A} \\ \emptyset \in \text{id} = \lambda x.x &\notin A \end{aligned} \quad \} \rightarrow \text{per Rice-Shapiro } A \text{ mom i.e.}$$

I due esercizi precedenti portano ad individuare due situazioni paradigmatiche per l'applicazione del teorema:

OSSERVAZIONE: Sia $A \subseteq \mathcal{C}$ un insieme di funzioni calcolabili.

I.e. $A = \{x \mid \varphi_x \in A\}$ è v.e.

Allora

(i) se $\forall \theta$ finita $\theta \notin A \Rightarrow A = \emptyset$

(ii) se $\emptyset \in A \Rightarrow A = \mathcal{C}$

dim

(i) data $f \in \mathcal{C}$, supponiamo che $f \in A$ se $\exists \theta$ finita $\theta \in A$ ma $f \notin A$

(ii) data $f \in \mathcal{C}$, dato che $\emptyset \subseteq f$ e $\emptyset \in A$ $\Rightarrow f \in A$ finita □

③ Studiare la ricorsività di $A = \{x \mid \varphi_x = \text{1}\}$

* A mom è v.e. ($\rightsquigarrow A$ mom ricorsivo)

infatti l'insieme di funzioni è in questo caso $A = \{1\}$ che

- mom contiene funzioni finite
 - mom è vuoto
- } $\rightarrow A$ mom v.e. per l'oss.
(punto (i))

* \bar{A} mom v.e.

infatti $\bar{A} = \mathcal{C} \setminus \{1\}$

si ha che

- $\emptyset \in \bar{A}$
- $\bar{A} \neq \mathcal{C}$ $\Rightarrow \bar{A}$ mom v.e. (oss. punto (ii))

OSSERVAZIONE: Il teorema di Rice-Shapiro dà una condizione necessaria, ma non sufficiente per essere t.c., ovvero non vale

$$\forall f \cdot (\text{f} \in A \text{ se e solo se } \exists \vartheta \text{ finito}, \vartheta \in f \text{ e } \vartheta \in A) \quad \Rightarrow \quad A \text{ t.c.}$$

(*)

Idea: il fatto che $f \in A$ si deduce da una parte finita di f non è sufficiente per concludere che A è t.c. dato che tale parte finita può non essere effettivo.

3

Rice-Shapiro si può usare per dim. che un insieme non è t.c.
non per dim. che un insieme è t.c.

Controesempio:

$$\text{sia } A = \{ f \in \mathbb{I} \mid \text{dom}(f) \cap \bar{K} \neq \emptyset \}, \quad A = \{ \vartheta \mid \varphi_\vartheta \in \mathcal{A} \}$$

* A soddisfa (*).

$$\begin{aligned} - \text{ se } f \in A &\Rightarrow \text{dom}(f) \cap \bar{K} \neq \emptyset \Rightarrow \\ &\Rightarrow \text{dallo } x \in \text{dom}(f) \cap \bar{K} \text{ si ha che } \vartheta = \{(x, f(x))\} \\ &\text{è finita, } \vartheta \in f \text{ e } \vartheta \in \text{dom}(\vartheta) \cap \bar{K} = \{x\} \neq \emptyset \end{aligned}$$

$$\begin{aligned} - \text{ se } \vartheta \text{ finita, } \vartheta \subseteq f, \vartheta \in \mathcal{A} &\Rightarrow \\ &\Rightarrow \text{essendo } \vartheta \subseteq f \quad \text{dom}(\vartheta) \subseteq \text{dom}(f) \\ &\Rightarrow \text{dom}(f) \cap \bar{K} \supseteq \text{dom}(\vartheta) \cap \bar{K} \neq \emptyset \\ &\Rightarrow f \in A \quad \text{In quanto } \vartheta \in \mathcal{A} \end{aligned}$$

* A non t.c. in quanto $\bar{K} \leq_m A$

Idea: per stabilire se $x \in \bar{K}$ guardo se $\{(x, 0)\} \in A$

definiamo

$$g(x, y) = \begin{cases} 0 & x = y \\ 1 & \text{altrimenti} \end{cases} = \mu z. |x - y|$$

è calcolabile, dunque per il th. smm \exists s. $\mathbb{N} \rightarrow \mathbb{N}$ colc. totale t.c.

$$g(x, y) = \varphi_{s(x)}(y)$$

dunque $\text{dom}(\varphi_{s(x)}) = \{x\}$ e quindi

$$\begin{aligned} - x \in \bar{K} &\Rightarrow \text{dom}(\varphi_{s(x)}) \cap \bar{K} = \{x\} \neq \emptyset \Rightarrow s(x) \in A \\ - x \notin \bar{K} &\Rightarrow \text{dom}(\varphi_{s(x)}) \cap \bar{K} = \emptyset \Rightarrow s(x) \notin A \end{aligned}$$

I° TEOREMA di RICORSIONE

Nei linguaggi di programmazione possiamo avere funzioni che hanno come argomenti delle altre funzioni.

Esempio in ML, la funzione succ che data una funzione f restituisce $f+1$ può essere definita da

$$\text{fun succ } f \ x = f\ x + 1;$$

Dal punto di vista della calcolabilità è dunque naturale chiedersi come possano essere caratterizzate le operazioni effettive/calcolabili su funzioni.

\Rightarrow questo porta alla nozione di funzionale ricorsivo

Def Indichiamo con $\mathcal{Y}(\mathbb{N}^k)$ l'insieme di tutte le funzioni (calcolabili e non) di k argomenti $\mathbb{N}^k \rightarrow \mathbb{N}$

Un funzionale è semplicemente una funzione

$$\Phi : \mathcal{Y}(\mathbb{N}^k) \rightarrow \mathcal{Y}(\mathbb{N}^h)$$

(qui consideriamo solo funzionali totali)

* Quando possiamo dire che un funzionale è effettivo, ovvero int. calcolabile

dato $\Phi : \mathcal{Y}(\mathbb{N}^k) \rightarrow \mathcal{Y}(\mathbb{N}^h)$ funzionale

- una funzione $f \in \mathcal{Y}(\mathbb{N}^k)$ e la sua immag. $\Phi(f) \in \mathcal{Y}(\mathbb{N}^h)$
sono oggetti infiniti (in generale)

- non possiamo chiedere che $\Phi(f)$ sia determinabile in un tempo finito da f

- si chiede che

$$\exists f \forall x$$

$\Phi(f)(x)$ effettivamente determinabile (in un tempo finito)

utilizzando solo una parte finita di f

(ovvero i valori che f assume su di un numero finito di argomenti)

Idea: questo si formalizza richiedendo che

$$\forall f \forall x \quad \Phi(f)(x) \text{ si ottenga dall'elaborazione effettiva di } \delta f \text{ finito}$$

Abbiamo dunque bisogno di vedere le funzioni finite come numeri; consideriamo dunque

* Codifica delle funzioni finite: per ogni funzione finita \mathcal{D} la sua codifica

$$\mathcal{D} \in \mathbb{N} \text{ è df se}$$

$$\text{- se } \mathcal{D} = \emptyset \quad \rightsquigarrow \quad \tilde{\mathcal{D}} = 0$$

$$\text{- se } \mathcal{D} = \{(x_1, y_1), \dots, (x_m, y_m)\} \quad \rightsquigarrow \quad \tilde{\mathcal{D}} = \prod_{i=1}^m p_x^{y_i+2}$$

mettiva e
effettiva
(quasi binario)
solo 1 mani immagine)

data la codifica di una funzione $\mathcal{Z} = \mathcal{D}$

$$x \in \text{dom}(\mathcal{D}) \quad \text{se} \quad (\mathcal{Z})_x \neq 0$$

$$\text{app}(\mathcal{Z}, x) = \mathcal{D}(x) = \begin{cases} (\mathcal{Z})_x = 1 & x \in \text{dom}(\mathcal{D}) \\ 1 & \text{altrimenti} \end{cases}$$

\Rightarrow in realtà così anche
 $\mathcal{Z} = 1$ codifica $\mathcal{D} = \emptyset$

$$= ((\mathcal{Z})_x = 1) \wedge (\mu w. \top \wedge (\mathcal{Z})_x \mid)$$

Così possiamo dare la seguente df.

Def (Funzionale Ricorsivo)

Un funzionale $\Phi: \mathfrak{I}(\mathbb{N}^k) \rightarrow \mathfrak{I}(\mathbb{N}^k)$ si dice RICORSIVO se esiste $\varphi: \mathbb{N}^{k+2} \rightarrow \mathbb{N}$ calcolabile t.c. $\forall f \in \mathfrak{I}(\mathbb{N}^k) \quad \forall z \in \mathbb{N}^k \quad \forall y \in \mathbb{N}$

$$\Phi(f)(y) = y \quad \text{sse} \quad \exists \mathcal{D} \subseteq f \text{ finita t.c. } \varphi(\mathcal{D}, \bar{x}) = y$$

si metti che φ può non essere totale

Esempio: Il funzionale

$$\text{fib} : \mathfrak{I}(\mathbb{N}) \rightarrow \mathfrak{I}(\mathbb{N})$$

$$\text{fib}(f)(x) = \begin{cases} 1 & \text{se } x = 0 \text{ o } x = 1 \\ f(x-2) + f(x-1) & \text{se } x \geq 2 \end{cases}$$

è ricorsivo; la funzione $\varphi: \mathbb{N}^2 \rightarrow \mathbb{N}$ può essere

$$\varphi(z, x) = \begin{cases} 1 & \text{se } x = 0 \vee x = 1 \\ \vartheta(x-2) + \vartheta(x-1) & \text{se } x \geq 2 \text{ e } z = \tilde{\vartheta} \quad (\tilde{\vartheta} \text{ indifinito anche } x = 1!) \end{cases}$$

$$= \begin{cases} 1 & \text{se } x = 0 \vee x = 1 \\ \text{opp}(z, x-2) + \text{opp}(z, x-1) & \text{se } x \geq 2 \end{cases}$$

calcolabile (con la solita tecnica...)

Esempio: Il funzionale associato alla funzione di Ackermann.

$$\Psi_{\text{Ack}}: \mathcal{T}(\mathbb{N}^2) \rightarrow \mathcal{T}(\mathbb{N}^2)$$

$$\Psi_{\text{Ack}}(f)(0, y) = y+1$$

$$\Psi_{\text{Ack}}(f)(x+1, 0) = f(x, 1)$$

$$\Psi_{\text{Ack}}(f)(x+1, y+1) = f(x, \Psi_{\text{Ack}}(f)(x+1, y))$$

come prima.

A conferma della appropriatezza dello df di funzionale ricorsivo, si può verificare che un funzionale ricorsivo trasforma funzioni calcolabili in funzioni calcolabili.

Teorema: Sia $\Phi: \mathcal{T}(\mathbb{N}^k) \rightarrow \mathcal{T}(\mathbb{N}^n)$ un funzionale ricorsivo e sia $f \in \mathcal{T}(\mathbb{N}^k)$ calcolabile. Allora $\Phi(f) \in \mathcal{T}(\mathbb{N}^n)$ è calcolabile.

* Teoremi di Myhill - Shepherdson

Dato un funzionale ricorsivo Φ , per il teorema appena visto:

$$f \text{ calcolabile} \rightsquigarrow \Phi(f) \text{ calcolabile}$$

$$f = \varphi_e \quad \Rightarrow \quad \Phi(f) = \varphi_e'$$

dunque possiamo vedere un funzionale ricorsivo come una funzione che trasforma indici (programmi) in indici (programmi) ma con la proprietà che la trasformazione dipende dalla funzione corrispondente all'indice, non dell'indice.

dipende dal significato (estensione) e non dal significante (intensione).

* funzione estensionale:

Una funzione totale $h: \mathbb{N} \rightarrow \mathbb{N}$ si dice estensionale se

$$\forall e, e' \quad \varphi_e = \varphi_{e'} \Rightarrow f(h(e)) = f(h(e'))$$

intuitivamente h dipende dalla funzione e non dall'indice

o induce una funzione su funzioni calcolabili

$$\Phi_h(\varphi_e) = \varphi_{h(e)} \quad \text{ben d'facto non dipende dal rappresentante}$$

Teorema di Myhill - Shepherdson (PART I)

Sia $\Phi : \mathcal{Y}(\mathbb{N}^k) \rightarrow \mathcal{Y}(\mathbb{N}^k)$ un funzionale ricorsivo. Allora esiste una funzione calcolabile totale $h_\Phi : \mathbb{N} \rightarrow \mathbb{N}$ t.c.

$$\forall e \quad \Phi(\varphi_e) = \varphi_{h_\Phi(e)}^{(k)}$$

Intuitivamente, il comportamento di un funzionale ricorsivo sulle funzioni calcolabili è catturato da una funzione totale estensionale su indici.

Teorema di Myhill - Shepherdson (PART II)

Sia $h : \mathbb{N} \rightarrow \mathbb{N}$ una funzione calcolabile totale estensionale. Allora

Allora esiste un unico funzionale ricorsivo Φ_h t.c.

$$\forall e \quad \Phi_h(\varphi) = \varphi_{h(e)}$$

Ovvero, una funzione calcolabile totale estensionale individua univocamente un funzionale ricorsivo.

E' interessante osservare che Φ_h è determinato anche sulle funzioni non calcolabili.

Poiché? Il comportamento sulle funzioni calcolabili determina anche quello sulle funzioni non calcolabili.

int.: ogni funzione è il "limite" delle sue sottofunzioni finite

$$f = \bigcup_{\text{def}} g$$

um funzionale ricorsivo è continuo

$$\Phi(f) = \bigcup_{\text{def}} \Phi(g)$$

e ogni funzione finita è calcolabile

Infine

I teorema di ricorsione (Kleene)

Sia $\Phi : \mathcal{Y}(\mathbb{N}^k) \rightarrow \mathcal{Y}(\mathbb{N}^k)$ un funzionale ricorsivo.

Allora Φ ha un minimo punto fisso f_Φ che è computabile, ovvero

$\exists f_\Phi \in \mathcal{Y}(\mathbb{N}^k)$ computabile t.c.

$$\cdot \Phi(f_\Phi) = f_\Phi$$

$$\cdot \forall g \in \mathcal{Y}(\mathbb{N}^k) \quad \Phi(g) = g \Rightarrow f_\Phi \subseteq g$$

e si può vedere che $f_\Phi = \bigcup_m \Phi^m(\emptyset)$

Dato che uno schema ricorsivo può essere sempre associato ad un punto fisso, il teorema mostra la chiusura dell'insieme delle funzioni calcolabili rispetto a forme di ricorsione estremamente generali (ogni forma "ragionevole")

Come così pochi colori abbiamo:

(1) Ricorsione primitiva

Date $f: \mathbb{N}^h \rightarrow \mathbb{N}$ e $g: \mathbb{N}^{h+2} \rightarrow \mathbb{N}$, la funzione definita per ric. prum.

è il minimo punto fisso di

$$\Phi_f \in \mathcal{Y}(\mathbb{N}^{h+1})$$

def. da

$$\Phi_f(h)(\bar{x}, 0) = f(\bar{x})$$

$$\Phi_f(h)(\bar{x}, y+1) = g(\bar{x}, y, h(\bar{x}, y))$$

Si può vedere che se f, g sono calcolabili, Φ_f è un funzionale ricorsivo

$$\varphi: \mathbb{N}^{k+2} \rightarrow \mathbb{N}$$

$$\varphi(z, \bar{x}, 0) = f(\bar{x})$$

$$\varphi(z, \bar{x}, y+1) = g(\bar{x}, y, \text{app}(z, \bar{x}, y))$$

quindi il teorema ci assicura che

- esiste un minimo punto fisso
- è calcolabile

(2) Minimalizzazione

Dato una funzione $f: \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ possiamo vedere la minimalizzazione

$$\mu y. f(\bar{x}, y)$$

come punto fisso

Consideriamo per f fissata

$$\Phi_f \in \mathcal{Y}(\mathbb{N}^{k+1})$$

$$\Phi_f(h)(\bar{x}, y) = \begin{cases} y & \text{se } f(\bar{x}, y) = 0 \\ h(\bar{x}, y+1) & \text{se } f(\bar{x}, y) < \epsilon \text{ e } f(\bar{x}, y) \neq 0 \\ \uparrow & \text{altrimenti} \end{cases}$$

è ricorsivo!

ha un minimo punto fisso $f_{\bar{x}}$ e si può vedere che

$$f_{\bar{x}}(\bar{x}, 0) = \mu y. f(\bar{x}, y)$$

più in generale

$$f_{\lambda}(x, y) = \mu z \cdot y \cdot f(z, y)$$

(3) Funzione di Ackermann

Abbiamo visto che Φ_{λ} è ricorsiva, dunque ha un minimo punto fisso (la funzione di Ackermann ψ) calcolabile.

Il fatto, dim. indipendentemente, che ψ sia totale implica che è l'unico punto fisso (i funzioni totali sono massimali)

(4) OSSERV.: Il punto fisso in generale non è unico, ad es.

$$\Phi(f)(x) = \begin{cases} 0 & \text{se } x=0 \\ f(x+1) & \text{altrimenti} \end{cases}$$

è ricorsivo e ha minimo punto fisso

$$f_{\Phi}(x) = \begin{cases} 0 & x=0 \\ 1 & \text{altrimenti} \end{cases}$$

ma ha molti altri punti fissi, ad es.:

$$f(x) = \begin{cases} 0 & x \leq 0 \\ k & x > 0 \end{cases}$$

SECONDO TEOREMA di Ricorsione

Sia h una funzione computabile totale estensionale. Allora per il teorema di Myhill - Shepherdson (part II) esiste un unico funzionale ricorsivo Φ tale che

$$\forall e \quad \Phi(\varphi_e) = \varphi_{h(e)}$$

Ora, poiché Φ è ricorsivo, per il I Teor di Ricorsione Φ ha un minimo punto fisso f_Φ calcolabile, dunque esiste $e \in \mathbb{N}$ (c. $f_\Phi = \varphi_e$) e pertanto

$$\varphi_e = \Phi(\varphi_e) = \varphi_{h(e)}$$

Orbene se h è computabile totale estensionale allora esiste m t.c.

$$\varphi_m = \varphi_{h(m)}$$

Il secondo teorema di ricorsione asserisce che questo vale anche se h non è estensionale.

Cosa significa?

Una funzione $h: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale (possibilmente non estensionale) non può essere pensata come funzionale su funzioni computabili

$$\Phi(\varphi_e) = \varphi_{h(e)} \quad \text{NON è bene definito}$$

dipende dall'indice e

ovvero se $\varphi_e = \varphi_{e'}$ può essere

$$\Phi(\varphi_e) = \varphi_{h(e)} \neq \varphi_{h(e')} = \Phi(\varphi_{e'})$$

Invece la possiamo pensare come una funzione che opera su programmi

$$f^*(P_e) = P_{h(e)}$$

ovvero come una trasformazione effettiva / modifica /azione di programmi
(cambia una istruzione, aggiungi una riga, etc.)

Il secondo teorema di ricorsione può dunque essere interpretato nel modo seguente

"dato un qualsiasi procedimento effettivo per trasformare programmi (un programma che opera su programmi)

esiste sempre un programma che una volta modificato fa esattamente quello che faceva prima (calcola la stessa funzione)"

detto un algoritmo.

« E' impossibile scrivere un programma che modifichi, in modo sostanziale, tutti i programmi. »

II Teorema di Ricorsione:

Sia $h: \mathbb{N} \rightarrow \mathbb{N}$ una funzione calcolabile totale. Allora esiste $m \in \mathbb{N}$ t.c.

$$\varphi_{h(m)} = \varphi_m$$

dim

consideriamo la funzione

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

3

con abuso di notazione si intende che se $\varphi_x(x) \uparrow$ allora

$$\varphi_{h(\varphi_x(x))} = \emptyset \text{ è la funzione sempre indeterminata}$$

chiaramente $g(x, y)$ è calcolabile.

$$g(x, y) = \varphi_0(h(\varphi_0(x, x)), y)$$

dunque per il teorema sussiste $s: \mathbb{N} \rightarrow \mathbb{N}$ calcolabile totale t.c.

$$g(x, y) = \varphi_{s(x)}(y) \quad \forall y$$

essendo s calcolabile $\exists m$ t.c. $s = \varphi_m$ e quindi sostituendo

$$\varphi_{h(\varphi_x(x))}(y) = \varphi_{\varphi_m(x)}(y) \quad \forall y$$

pertanto, per $x = m$

$$\varphi_{h(\varphi_m(m))}(y) = \varphi_{\varphi_m(m)}(y) \quad \forall y$$

ovvero posto $m = \varphi_m(m)$

(definito in quanto $s = \varphi_m$ è totale)

$$\varphi_{h(m)}(y) = \varphi_m(y) \quad \forall y$$

ossia

$$\varphi_{h(m)} = \varphi_m$$

Nota: La dimostrazione appare oltranzista e misteriosa, ma si tratta di un semplice argomento diafumale.

Una funzione calcolabile totale fornisce una enumerazione di un sottoinsieme di funzioni calcolabili

$$\varphi_{f_0}, \varphi_{f_1}, \varphi_{f_2}, \dots \quad (*)$$

e quando f non sia totale, con la convenzione che se $f(m) \uparrow$ allora $\varphi_{f(m)} = \phi$, possiamo comunque considerare l'enumerazione $(*)$.

Dunque l'enumerazione delle funzioni calcolabili

$\varphi_0, \varphi_1, \varphi_2, \dots$ fornisce una enumerazione delle possibili enumerazioni effettive

$$\text{Eo (enum. indotta da } \varphi_0\text{)} \quad \varphi_{\varphi_0(0)}, \varphi_{\varphi_0(1)}, \varphi_{\varphi_0(2)}, \dots$$

$$\text{Ei (" " " " } \varphi_1\text{)} \quad \varphi_{\varphi_1(0)}, \varphi_{\varphi_1(1)}, \varphi_{\varphi_1(2)}, \dots$$

$$\text{Em (" " " " } \varphi_m\text{)} \quad \varphi_{\varphi_m(0)}, \varphi_{\varphi_m(1)}, \varphi_{\varphi_m(2)}, \dots, \varphi_{\varphi_m(m)}$$

Ora, data una funzione calcolabile totale h

$$\varphi_{h(\varphi_0(x))} \sim \varphi_{h(\varphi_0(0))}, \varphi_{h(\varphi_0(1))}$$

è una enumerazione effettiva di funzioni calcolabili, indotta da

$$h(\varphi_0(x)) = h(\varphi_0(x, x))$$

e dunque deve coincidere con qualche E_m : no (vedi figura) c'è una interazione

$$\varphi_{\varphi_0(0)}$$

$$\varphi_{\varphi_1(1)}$$

$$\text{Eo } \varphi_{h(\varphi_0(0))}, \varphi_{h(\varphi_0(1))}, \dots, \varphi_{h(\varphi_m(m))} = \varphi_{\varphi_m(m)}$$

ovvero

$$\varphi_{\varphi_m(m)} = \varphi_{h(\varphi_m(m))}$$

il cui pb. potrebbe essere che $\varphi_m(m) \uparrow$, ma si può facilmente verificare che per ogni f calcolabile esiste f' calcolabile totale che induce la stessa enumerazione.

basta considerare

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

$$= \varphi_0(f(x), y)$$

e per il teorema sull'insieme f calcolabile

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

$\rightsquigarrow f'$ è la funzione desiderata

Il secondo teorema di ricorsione è un risultato estremamente profondo, così molti risultati visti finora seguono come semplici corollari di esso.

* Teorema di Rice. Sia $\phi \neq A \in \mathbb{N}$. Se A saturoto $\Rightarrow A$ non ricorsivo.

dim

sia $\phi \neq A \in \mathbb{N}$

supponiamo per assurdo A (e quindi \bar{A}) ricorsivo

$\rightsquigarrow x_A$ e $x_{\bar{A}}$ calcolabili

siamo $a_0 \in A$

elementi fissati (esistono certamente in quanto A non vuoto)

$a_1 \in A$

e definiammo

$$f(x) = \begin{cases} a_0 & x \in A \\ a_1 & x \notin A \end{cases} = a_0 x_A(x) + a_1 x_{\bar{A}}(x)$$

è facile vedere che

$$x \in A \iff f(x) \in \bar{A} \quad (*)$$

Ora f è computabile totale, quindi per il II teor di ricorsione

$\exists m \in \mathbb{N} : f(m) \in \bar{A}$

\rightsquigarrow membro A saturoto m , $f(m) \in A$ oppure $m, f(m) \in \bar{A}$

il che contraddice $(*)$

assurdo!

* K_{mom} è ricorsivo

dim

sia per omurdo $K = \{x \mid x \in \mathbb{W}_x\}$ ricorsivo

e siamo e_0, e_1 indici t.c. $\varphi_{e_0} = \emptyset$ e $\varphi_{e_1} = \lambda x x$

definiamo

$$f(x) = \begin{cases} e_0 & x \in K \\ e_1 & x \notin K \end{cases}$$

f è computabile totale $\Rightarrow x$ il II teor. di ricorsione $\exists m \quad \varphi_m = \varphi_{f(m)}$

ma

$$\forall e \in K \Rightarrow \varphi_m = \varphi_{f(m)} = \varphi_{e_0} \Rightarrow \varphi_m(m) = \varphi_{e_0}(m) \uparrow \Rightarrow m \notin K$$

$$\forall m \notin K \Rightarrow \varphi_m = \varphi_{f(m)} = \varphi_{e_1} \Rightarrow \varphi_m(m) = \varphi_{e_1}(m) = m \Rightarrow m \in K$$

omurdo!

□

* Se A è sottratto $A \not\leq_m \bar{A}$

dim

se fosse $A \leq_m \bar{A}$ visterebbe f calcolabile totale t.c.

$$x \in A \iff f(x) \in \bar{A}$$

3

$\forall m \quad \varphi_m \neq \varphi_{f(m)}$ omurdo (contraddice il II teor di ricorsione)

* $\exists m_0 \text{ t.c. } \varphi_{m_0} = f(m_0, m_0)$

(ovvero $\varphi_{m_0}(x) = \begin{cases} m_0 & x = m_0 \\ \uparrow & \text{altrimenti} \end{cases}$)

dim

consideriamo

$$g(m, x) = \begin{cases} m & x = m \\ \uparrow & \text{altrimenti} \end{cases}$$

$$= m \cdot \mathbb{1}_{\{x = m\}}$$

calcolabile e quindi per il th. s.m.m. è s.n.-in calc totale t.c.

$$g(m, x) = \varphi_{S(m)}(x)$$

$$\text{ovvero } \varphi_{S(m)} = \{(m, m)\}$$

Essendo S calc totale, per il II teorema di ricorsione vi è $m_0 \in \mathbb{N}$

t.c.

$$\varphi_{m_0} = \varphi_{S(m_0)} = \{(m_0, m_0)\}$$

come disidurato

□

Graffito: K mom i saturato

dinner

$$\text{Sia } m_0 \text{ tc: } \varphi_{m_0} = \varphi(m_0, m_0)$$

2

$\alpha_0 \in K$

E' moto che esistono infiniti indici per una funzione; sia quindi $m \neq m'$

L.C. $P_m = P_{mo}$ \rightarrow k mom saturato

$$m \in \varphi_m(m) \uparrow \Rightarrow m \notin K$$