Parte I

Computabilità

Capitolo 1

Funzioni calcolabili e Modelli di calcolo

1.1 Introduzione

Ci sono dei problemi che non possono essere risolti in modo algoritmico, come la terminazione o la prova di correttezza di un programma, lo studio di questi problemi prende il nome di teoria della computabilità.

In questa teoria non viene preso in considerazione il consumo di risorse in modo che le dimostrazioni effettuate siano indipendenti dal modello di calcolo adottato.

Notoriamente, i problemi appartengono a varie classi di difficoltà:

- **P**: problemi che possono essere risolti da un algoritmo in tempo polinomiale
- **NP**: problemi che possono essere risolti in tempo polinomiale ma in modo non deterministico
- **EXP**: problemi che possono essere risolti da un algoritmo in tempo esponenziale

1.1.1 L'informatica e la computabilità

Computer science is no more about computers tha astronomy is about telescopes. Dijkstra.

L'idea dell'informatica nasce dalla logica, ricercando un procedimento generale (macchina) su base combinatoria per trovare tutte le verità.

Libro: *Nigel Cutland "Computability. An Introduction to Recursive Function Theory" Cambridge University Press.*

1.2 Algoritmo

Algoritmo: descrizione di una sequenza di passi elementari che permettono di raggiungere un certo obiettivo, come la trasformazione di certi dati di input in dati di output.

Il vincolo dei passi elementari si traduce nell'utilizzo di operazioni semplici, direttamente eseguibili da un calcolatore.

Nel caso l'algoritmo sia deterministico, questo può essere visto come una funzione che mappa un certo input in un determinato output, in questo caso si dice che la funzione **è calcolata** dall'algoritmo.

Funzione calcolabile: una funzione è calcolabile in modo effettivo se **esiste** un algoritmo in grado di calcolarla. A noi interessa sapere che **esiste** un algoritmo che calcola quella determinata funzione, ma non sempre ci interessa conoscere l'algoritmo.

Alcune funzioni calcolabili sono:

$$MCD(x, y)$$

$$P(n) = \begin{cases} 1, & \text{se } n \text{ è primo} \\ 0, & \text{altrimenti} \end{cases}$$

$$p(n) = n\text{-esimo numero primo}$$

$$f(n) = n\text{-esima cifra di } \pi$$

Ci sono però delle funzioni per le quali non si riesce a stabilire se sono calcolabili o meno, come per la funzione:

$$g(n) = \begin{cases} 1, & \text{se } \pi \text{ contiene esattamente } n \text{ cifre “5” consecutive} \\ 0, & \text{altrimenti} \end{cases}$$

Questo perché si riesce a trovare un algoritmo che funziona solamente nel caso in cui la sequenza è presente.

Ci sono poi altre funzioni che sembrano simili a *g(n)* ma che in realtà sono calcolabili, come la funzione:

$$h(n) = \begin{cases} 1, & \text{se } \pi \text{ contiene almeno } n \text{ cifre “5” consecutive} \\ 0, & \text{altrimenti} \end{cases}$$

È infatti possibile definire un algoritmo per il calcolo utilizzando il numero k

$$k = sup(\{n | \pi \text{contiene } n \text{ cifre “5” consecutive}\})$$

k può essere un numero qualsiasi o infinito. In ogni caso, la funzione h può essere definita come

$$h(n) = \begin{cases} 1, & \text{se } n \leq k \\ 0, & \text{altrimenti} \end{cases}$$

Da notare che l'algoritmo così definito non si preoccupa di quanto sia difficile trovare k, ma solo di trovare una soluzione per il problema. Questo deriva dall'**esiste** precedentemente riportato in grassetto.

Noi ci accontentiamo del fatto che per qualsiasi valore che può assumere k siamo in grado di definire un algoritmo. L'algoritmo per trovare k può esistere o meno, ma questo non ci interessa perché vogliamo che la teoria della calcolabilità non venga influenzata dalla conoscenza sul dominio ma che sia una caratteristica della funzione. Ad esempio, in questo caso non ci interessa sapere se il π è un numero normale o meno e non vogliamo che la definizione di calcolabilità sia influenzata da ciò.

Si potrebbe pensare di calcolare *g(n)* utilizzando una funzione analoga, utilizzando l'insieme X contenente tutte le sequenze di numeri presenti nel π

$$g'(n) = \begin{cases} 1, & \text{se } n \in X \\ 0, & \text{altrimenti} \end{cases}$$

ma l'insieme X può essere un insieme infinito e per valutare l'appartenenza è necessario un programma infinito.

1.3 Le funzioni non calcolabili

Modello di calcolo effettivo: l'insieme delle caratteristiche che deve avere un algoritmo per poter essere eseguibile in modo effettivo.

- **Lunghezza sintattica finita**: deve essere una sequenza finita di istruzioni.
- **Modello realistico**: esegue su un agente di calcolo che può essere realizzato, dotato di una memoria e che lavora a passi discreti (*macchina digitale*) e deterministici.

- **Memoria e input illimitati**: questo perché non si vuole limitare la teoria della computabilità allo stato tecnologico attuale, idealmente si può sempre aggiungere un nuovo banco di RAM.
- **Set di istruzioni finito**: le istruzioni che la macchina riesce ad eseguire sono limitate ed hanno anche una complessità limitata.
- **Computazione**: può terminare dopo un numero finito di passi producendo un output, oppure può non terminare senza produrre output.

Questo modello equivale ad una macchina di Turing.

1.3.1 Notazione utilizzata

Notazioni:

- $\mathbb{N} = \{0, 1, 2, 3, \dots\}$
- $A \times B = \{(a, b) | a \in A, b \in B\}$ (prodotto cartesiano)
- $A \times A \times \dots \times A = A^n$
- Relazione $r \subseteq A \times B$, ovvero un sotto insieme del prodotto cartesiano di uno o più insiemi.
- Una funzione è una particolare relazione: $f \subseteq A \times B$ e vale anche $\forall (a, b), (a, b') \in f \Rightarrow b = b'$ e $f(a) = b$
- Dominio di una funzione: $dom(f) = \{a \mid \exists (a, b) \in f\}$ e per indicare che f è definita su a si utilizza $f(a) \downarrow$.
- Cardinalità $|A|$, numero di elementi presenti nell'insieme nel caso di elementi finiti, mentre nel caso di insiemi infiniti si ha che:
  - $|A| = |B|$ se esiste $f : A \to B$ e biunivoca[1]
  - $|A| \leq |B|$ se esiste $f : A \to B$ e iniettiva[2]
- Un insieme è **numerabile** quando $|A| \leq |\mathbb{N}|$, ovvero esiste una funzione suriettiva $f : |\mathbb{N}| \to A$. Se A e B sono numerabili, anche il loro prodotto è numerabile. Allo stesso modo se una sequenza di insiemi è numerabile, anche la loro unione è enumerabile.

1.3.2 Esistenza delle funzioni non calcolabili

Con il modello di calcolo precedentemente descritto ci sono delle funzioni che non possono essere calcolate.

Fissando un insieme di funzioni unarie e parziali

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

e un modello di calcolo con i relativi algoritmi $\mathcal{A}$.

Le funzioni calcolabili nel modello di calcolo sono date da:

$$F_{\mathcal{A}} = \{f \mid \exists A \in \mathcal{A} \text{ che calcola } f\}$$

dato un algoritmo $A \in \mathcal{A}$ riusciamo a trovare una funzione f_A che viene calcolata.

La domanda risulta quindi essere "$F_{\mathcal{A}} \subseteq F$ o $F_{\mathcal{A}} \subsetneq F$?" e la risposta è che l'inclusione è stretta.

[1] Una funzione $f : X \to Y$ è biunivoca se e solo se ad ogni elemento di X è associato un solo elemento di Y e viceversa.

[2] Una funzione $f : X \to Y$ si dice iniettiva se due elementi distinti del dominio hanno immagini distinte, ovvero $a_1 \neq a_2$ implica $f(a_1) \neq f(a_2)$.

Dimostrazione Sia I il set finito di istruzioni della macchina di calcolo.

L'insieme degli algoritmi calcolabili risulta quindi essere:

$$\mathcal{A} \subseteq I \cup (I \times I) \cup (I \times I \times I) \cup \ldots = \bigcup_n I^n$$

pertanto

$$|\mathcal{A}| \leq |\bigcup_n I^n| \leq |\mathbb{N}|$$

Poiché la funzione $f : \mathcal{A} \to F_{\mathcal{A}}$ è ovviamente suriettiva[3], si ha:

$$|F_{\mathcal{A}}| \leq |\mathcal{A}| \leq |\mathbb{N}|$$

L'insieme delle funzioni è certamente infinito e certamente esiste un sottoinsieme T delle funzioni totali che non è enumerabile.

$$T = \{f | f : \mathbb{N} \to \mathbb{N} \text{ e totali}\}$$

La non enumerabilità si dimostra per assurdo, perché se $f_0,\ f_1, \ldots$ è un'enumerazione di T:

```
    f0       f1       f2       ...
0   f0(0)    f1(0)    f2(0)    ...
1   f0(1)    f1(1)    f2(1)    ...
2   ...      ...      ...
```

È quindi possibile definire $d(n) = f_n(n) + 1$ che non è presente nell'enumerazione sopra riportata, perché $d(n) \neq f(n) \forall n$.

Pertanto

$$|\mathbb{N}| \leq |T| \leq |F|$$

e

$$\left.\begin{array}{l} F_{\mathcal{A}} \subseteq F \\ |F_{\mathcal{A}}| < |F| \end{array}\right\} \Rightarrow\ F_{\mathcal{A}} \subsetneq F$$

Ovvero, esistono delle funzioni che non sono calcolabili. Il numero di queste funzioni è dato da $|F - F_{\mathcal{A}}|$ che è infinita, questo perché se fosse finita F sarebbe enumerabile.

1.4 Modelli di calcolo

Nel tempo sono stati proposti diversi modelli di calcolo:

- Macchina di Turing, 1936
- Lambda calcolo, Church 1936
- Funzioni parziali ricorsive, Göedel-Kleene
- Sistemi di Post, grammatiche libere da contesto
- Markov System, 1951
- Macchina a registri, 1963

La lista dei modelli è lunga, pertanto si può pensare che sia necessaria una teoria della computabilità per ogni modello. Tuttavia si è notato che le classi delle funzioni calcolabili dai vari modelli erano sempre quelle. Queste congetture sono state poi formalizzate dalla **tesi di Church-Turing**: ogni funzione è calcolabile con un procedimento effettivo se e solo se è calcolabile da una macchina di Turing.

[3]Una funzione $f : X \to Y$ è detta suriettiva se $\forall y \in Y, \exists x \in X | f(x) = y$.

1.4.1 URM - Ulimited Register Machine

Il modello di calcolo che utilizzeremo è quello della macchina URM, la quale è dotata di una serie di registri, ognuno dei quali contiene dei numeri naturali.

```
 R1 R2 R3
|r1|r2|r3|..|..|..
```

$$r_i \in \mathbb{N}$$

C'è inoltre un agente in grado di eseguire un programma, ovvero una sequenza di azioni $I_1, I_2, \ldots, I_n$. Queste istruzioni possono essere:

- Aritmetiche:
  - `Z(n)`: sposta 0 in r_n, $r_n \leftarrow 0$
  - `S(n)`: incrementa di 1 il valore del registro r_n, $r_n \leftarrow r_n + 1$
  - `T(m,n)`: copia il contenuto del registro r_m in r_n, $r_n \leftarrow r_m$.
- Salto condizionato:
  - `J(m,n,t)`: se r_m è uguale a r_n, salta all'istruzione I_t, altrimenti procede con l'istruzione successiva. Se viene fatto un salto ad un'istruzione che si trova fuori dal programma, l'esecuzione termina.

Un programma d'esempio è dato da:

```
I1: J(2,3,5)
I2: S(1)
I3: S(3)
I4: J(1,1,1)
```

l'esecuzione del programma risulta quindi essere:

```
   R1 R2 R3 R4
  |1 |2 |0 |0 |

I1: false, non salta

I2:  R1 R2 R3 R4
    |2 |2 |0 |0 |

I3:  R1 R2 R3 R4
    |2 |2 |1 |0 |

I4: true, esegue I1:

I1: false, non salta

I2:  R1 R2 R3 R4
    |3 |2 |1 |0 |

I3:  R1 R2 R3 R4
    |3 |2 |2 |0 |
```

`I4: true, esegue I1:`

`I1: true, salta alla terminazione`

La macchina ha memoria infinita e lo stato iniziale della computazione del programma P viene indicato con:

$$P(a_1, a_2, \ldots) \downarrow$$

La notazione

$$P(a_1, a2, \ldots) \downarrow a$$

indica che il programma P eseguito sulla configurazione $a_1, a_2, \ldots, a_n, 0, \ldots$ termina producendo in r_1 il valore a.

Per indicare che il programma non termina viene utilizzato:

$$P(a_1, a_2, \ldots) \uparrow$$

1.4.2 Funzioni URM-calcolabili

Una funzione parziale $f : \mathbb{N}^k \to \mathbb{N}$ si dice **URM-calcolabile** se esiste P programma URM tale che per ogni input $a_1, \ldots, a_k \in \mathbb{N}$, il programma P termina producendo $a \in \mathbb{N}$ se e solo se la funzione è definita sulla tupla e il risultato della funzione coincide con a.

Più formalmente:

$$P(a_1, a_2, \ldots, a_k) \downarrow \text{ calcola } f \text{ se } \begin{cases} (a_1, a_2, \ldots, a_k) \in dom(f) \\ f(a_1, a_2, \ldots, a_k) = a \end{cases}$$

Con la lettera $\mathcal{C}$ indichiamo la classe delle funzioni URM-calcolabili e con $\mathcal{C}^k$ ci si restringe alle funzioni k-arie. Un esempio di funzione calcolabile è dato da $f(x,y) = x+y$, la quale può essere calcolata dal programma:

```
/*
   R1 R2 R3 ...
  | x| y| 0|...
  uso r3 come k, e cerco di fare in modo che r1 sia uguale a x+k
*/
I1: J(2,3,END) #LOOP
I2: S(1) // x = x + 1
I3: S(3) // k = k + 1
I4: J(1,1,LOOP)
#END
```

Un altro esempio è dato da $f(x) = \begin{cases} \frac{x}{2} & \text{se } x \text{ è pari} \\ \uparrow & \text{altrimenti} \end{cases}$

```
/*
  R1 R2 R3 ...
  | x| 0| 0|...

  uso r2 come k e r3 come 2k.
  Incremento 2k di 2 e k di 1, quando 2k = x, allora k è x/2
*/
I1: J(1,3,END) #LOOP
I2: S(2) // k = k + 1
I3: S(3) // 2k = 2k + 1
I3: S(3) // 2k = 2k + 1
```

```
I4: J(1,1,LOOP)
I5: T(2,1) #END
```

Da notare che per individuare la funzione calcolata da un programma URM è necessario specificare sia il programma che il numero di argomenti, altrimenti non è possibile definire l'arietà della funzione.

$$f_p^{(k)} : \mathbb{N}^k \to \mathbb{N}$$

$$f_p(\vec{x}) = \begin{cases} y & \text{se } P(\vec{x}) \downarrow y \\ \uparrow & \text{se il programma non termina} \end{cases}$$

Quando una funzione è calcolabile, esistono infiniti programmi in grado di calcolarla, perché risulta semplice andare ad aggiungere delle istruzioni "inutili" per generare un nuovo programma che calcola la funzione, mentre se non è calcolabile, non esiste nessun programma in grado di calcolarla.

Esercizio pratico - Programma in URM

Implementare *f(x) = x-1, f(0) = 0*.

Soluzione Prima di tutto controllo se $x = 0$, in questo caso il programma termina ritornando 0, altrimenti eseguo la sottrazione.

Per modellare la sottrazione utilizzo due registri, uno che parte da 0 e uno che parte da 1. Ad ogni passo sommo 1 ad entrambi i registri e quando il registro che inizialmente conteneva 1 è uguale ad x, nell'altro registro ho il valore *x-1*.

```
/*
R1 R2 R3 ...
| x| 0| 0|...
r2 = k
r3 = k-1
*/
J(1,3,END) // x = r3= 0 -> fine ritornando 0
S(2) // Inizializzo r2 a 1
J(1,2,END) #LOOP
S(2) //k++
S(3) //(k-1)++
 J(1,1,LOOP)
T(3,1) #END // ritorna x-1 (0 se x = 0)
```

Esercizio teorico - URM e URM'

Sia URM' la macchina senza `T(m,n)` e sia $\mathcal{C}'$ la classi di funzioni URM'-calcolabili, questa classe come si relaziona con $\mathcal{C}$? Come prima cosa ci si può chiedere se

$$\mathcal{C}' \subseteq \mathcal{C}$$

Sia $f : \mathbb{N}^k \to \mathbb{N}$ URM' calcolabile, ovvero $f \in \mathcal{C}'$, allora esiste P' programma in URM' tale che $f = f_p$. Per come è definito il set di istruzione URM', P' è anche un programma in URM, quindi $f \in \mathcal{C}$

Resta ora da capire se l'inclusione è stretta o meno. L'istruzione `T(m,n)` può essere sostituita dalle istruzioni

```
Z(n)
J(n,m,END) #LOOP
S(n)
```

```
J(1,1,LOOP)
#END
```

quindi, essendoci un modo per simulare l'istruzione del trasferimento si ottiene che

$$\mathcal{C} \subseteq \mathcal{C}'$$

e quindi le due classi **sono equivalenti**. Tuttavia, questa è una congettura e non una dimostrazione formale.

Dimostrazione formale Sia $f \in \mathcal{C}$, ovvero esiste un P in URM e $k \in \mathbb{N}$ tale che $f = f_p^{(k)}$, si vuole dimostrare che esiste un programma P' in URM', tale che $f_{p'}^{(k)} = f_p^{(k)}$. Questo può essere dimostrato per induzione su h = *#numero di istruzioni* `T(m,n)` *in* P.

Banalmente se h=*0* si ha che P è già un programma URM', mentre se $h > 0$, il programma P avrà un'istruzione I_s che sarà un trasferimento `T(m,n)`. Tale programma programma può essere riscritto come P'', con I_s uguale a `J(1,1,l+2)`, dove all'istruzione I_{l+2} iniziano le istruzioni alternative per fare la copia del valore, come precedentemente riportato, e al termine delle quali c'è un salto all'istruzione I_{s+1}

Resta un problema riguardo ad i salti oltre la fine, è quindi necessario normalizzare i salti in modo che tutti quelli che puntano al termine del programma finiscano all'istruzione I_{l+1}, dove I_l è l'ultima istruzione del programma P. L'istruzione I_{l+1} di P'' sarà quindi un salto all'effettiva fine del programma.

Applicando questo procedimento si ottiene un programma P'' che avrà un'istruzione `T(m,n)` in meno e che calcola la stessa funzione calcolata da P. Ripetendo questo processo si ottiene induttivamente il programma P' che non ha istruzioni del tipo `T(m,n)`.

Esercizio teorico - URM e URM-S

Come si relazione la classe $\mathcal{C}^S$ con $\mathcal{C}$? Dove $\mathcal{C}^S$ è la classe delle funzioni calcolabili da URM-S ovvero dalla macchina URM che non ha l'istruzione `T(m,n)` ma ha l'istruzione `TS(m,n)` che fa lo swap del contenuto di dei due registri.

Soluzione

$\mathcal{C} \subseteq \mathcal{C}^S$ Sia $f \in \mathcal{C} \Rightarrow$ esiste un programma P in URM tale che $f = f_P^{(k)}$, si vuole trovare P' in URM-S tale che $f_P^{(k)} = f_{P'}^{(k)}$.

Questo può essere fatto mediante induzione sul numero di istruzioni `T` presenti in P, mostrando come queste possano essere trasformate in istruzioni `TS`, oppure si può sfruttare quanto dimostrato nell'esercizio della sezione §**??**, ovvero che la classe delle funzioni calcolate senza la funzione di trasferimento $\mathcal{C}'$ è equivalente alla classe $\mathcal{C}$ e che sicuramente $\mathcal{C}' \subseteq \mathcal{C}^S$ perché un programma URM' è anche un programma URM-S senza istruzioni `TS`.

Si ha quindi che $\mathcal{C} \equiv \mathcal{C}' \subseteq \mathcal{C}^S$.

$\mathcal{C}^S \subseteq \mathcal{C}$ Sia $f \in \mathcal{C}^S \Rightarrow$ esiste un programma P in URM-S tale che $f = f_P^{(k)}$, si vuole trovare P' in URM tale che $f_P^{(k)} = f_{P'}^{(k)}$.

Dimostriamo quindi che dato P programma in URM-S, esiste P' in URM tale che $f_P^{(k)} = f_{P'}^{(k)}$ per induzione su h = *numero di istruzioni* `TS(m,n)` *in* P.

- $(h = 0)$, non ci sono istruzioni `TS` quindi P è già in URM.
- $(h \Rightarrow h + 1)$, ovvero il programma ha un certa istruzione s `TS(m,n)` e termina all'istruzione *l+1*.

Viene scelto un registro $t = \max(k, \#\text{registri usati da } p)$. Il programma P viene quindi modificato in modo che punti ad un blocco di codice del tipo

```
T(m,t)
T(n,m)
T(t,n)
J(1,1,s+1)
```

e viene anche modificata l'istruzione *l+1* in modo che faccia un salto alla fine del programma, ovvero *l+5*.

Si ottiene quindi *P"* con h istruzioni `TS`, quindi per induzione esiste *P'* in URM tale che $f_{P'}^{(k)} = f_{P''}^{(k)} = f_P^k$.

C'è un **piccolo problema** dato che al passo induttivo si ottiene un programma ibrido che contiene sia istruzioni di scambio, sia istruzioni di trasferimento.

Per sistemare questa sottigliezza è necessario modificare la dimostrazione:

"Dimostriamo che dato un programma P che contiene **sia istruzioni di scambio che istruzioni di trasferimento**, esiste *P'* ...".

Esercizio teorico - URM e URM-SL

Sempre quella domanda, con la differenza che viene tolta l'istruzione di salto `J(m,n,l)`.

Soluzione Ovviamente $\mathcal{C}^{SL}$ è contenuta in $\mathcal{C}$, ma non vale la relazione $\mathcal{C} \subseteq \mathcal{C}^{SL}$ perché $\mathcal{C}^{SL}$ contiene solamente funzioni totali, ovvero che terminano sempre, mentre $\mathcal{C}$ contiene anche delle funzioni parziali.

Per le istruzioni disponibili in URM-SL si riesce a calcolare *f(x) = k* oppure *f(x) = x+k* per $k \in \mathcal{N}$. Ovvero $f \in \mathcal{C}^{SL}$ se e solo se *f(x) = k oppure x+k*.

Questo vuol dire che per ogni programma P_{SL} la funzione che viene calcolata $f_{p_{SL}}$ è del tipo di *f*.

Dimostrazione La dimostrazione viene fatta per induzione sulla lunghezza del programma *h=L(P)*.

($h = 1$) in questo caso il programma ha solamente un'istruzione, la quale può essere:

- `Z(n)`: se n è uguale a 1, il programma azzera il primo registro, quindi calcola 0, altrimenti se azzera un registro qualsiasi, il programma non modifica il valore di input, quindi calcola $f_P(x) = x$.
- `S(n)`: in modo analogo a prima, se *n=1* il programma calcola $f_P(x) = x + 1$, altrimenti calcola $f_P(x) = x$.
- `T(m,n)`: se *n=1* e $m > 1$, il programma calcola 0, perché tutti i registri che non sono di input sono per definizione a 0. Se invece $n > 1$ oppure se *m=1*, $f_P(x) = x$.

($h \Rightarrow h + 1$) il programma P può essere visto come un programma composto dal programma *P'* e dell'istruzione *h+1*.

Per ipotesi induttiva si ha che $f_{P'}$ ha la forma desiderata, ovvero $f_{P'}(x) = k$ oppure $x + k$.

L'istruzione *h+1*, può essere:

- `Z(n)`: se n è uguale a 1, il programma azzera il primo registro, quindi calcola 0, altrimenti se azzera un registro qualsiasi il programma non modifica il valore attuale del registro, quindi calcola $f_P(x) = f_{P'}(x)$.

- `S(n)`: se *n=1* il programma calcola $f_P(x) = f_{P'}(x) + 1$, se $n > 1$ il registro non viene modificato quindi $f_P(x) = f_{P'}(x)$.
- `T(m,n)`: se $n > 1$ oppure *m=1* si ha $f_P(x) = f_{P'}(x)$. Se invece *n=1* e $m > 1$ cade il palco, perché non si sa niente sul contenuto del registro m. È necessario quindi **modificare la dimostrazione** che si vuole fare in $f_p^{(1)(t)}(x) = k$ o $x + k \, \forall t$. La dimostrazione è molto simile a quella svolta finora, con la differenza che in questo caso si ha la garanzia che nel registro m ci sia un valore delle forma desiderata.

Con la modifica indicata è possibile assumere che anche negli altri registri ci può essere solamente x oppure $x+k$, rendendo così possibile dimostrare che anche nel caso in cui $m > 1$ il valore che viene calcolato è di quella forma.

1.5 Predicati decidibili

Nel mondo matematico spesso ci si chiede se valgono determinate proprietà come:

$$div(n, m) = \exists k : n \cdot k = m$$

La stessa cosa può essere vista come una relazione sui numeri naturali $div \subseteq \mathbb{N} \times \mathbb{N}$. Tipicamente questo prende il nome di **problema** o **predicato** e viene formalmente indicato con

$$Q(x_1, \ldots, x_k)$$

$$Q : \mathbb{N}^k \to \{vero, falso\} \text{ oppure } Q \subseteq \mathbb{N}^k$$

Una volta formalizzato il predicato, ci si chiede se questo predicato è **calcolabile** o **decidibile** Un predicato $Q \subseteq \mathbb{N}^k$ si dice **decidibile** quando $\mathcal{X}_A : \mathbb{N}^k \to \mathbb{N}$ è calcolabile, ovvero

$$\mathcal{X}(\vec{x}) = \begin{cases} 1, & \text{se } Q(\vec{x}) \\ 0, & \text{altrimenti} \end{cases}$$

Un predicato è decidibile quando la sua funzione $\mathcal{X}_Q$ è totale.

1.5.1 Esempi di predicati decidibili

Uguaglianza

Ad esempio $Q_1(x, y) =$ "$x = y$" (uguaglianza) viene calcolato da

$$\mathcal{X}_{Q_1}(x, y) = \begin{cases} 1, & \text{se } x = y \\ 0, & \text{altrimenti} \end{cases}$$

Risulta triviale trovare un programma che calcola questa funzione

```
J(1,2,SI)
Z(1)
J(1,1,FINE)
Z(1) #SI
S(1)
#FINE
```

Pari e dispari

Altro esempio $Q_2(x) =$ "x è pari"

```
/*
    1: x
    2: 0, lo incremento per vedere se x è pari o meno
    3: 0, lo uso per tenere il codice compatto
*/
J(1,2,SI) #PARI?
S(2)
J(1,2,NO) #DISPARI?
S(2)
J(1,1,PARI?)
S(3) #SI
T(3,1) #NO
```

Esercizio - Minore e uguale

$Q_3(x, y) = x \leq y$

todo

Esercizio - Divisibile

$Q_4(x, y) = div(x, y)$

todo

1.6 Computabilità su altri domini

Ci sono tanti modelli di calcolo che funzionano su svariati domini e noi ci siamo concentrati solamente su quelli che utilizzano i numeri. Può però essere interessante vedere come cambiano le regole del gioco se viene utilizzato un dominio diverso.

Codifica effettiva: Dato un dominio D, si può trovare una funzione $\alpha : D \to \mathbb{N}$ biunivoca e effettiva, con inversa effettiva.

Alcuni domini che vanno bene sono $\mathbb{Z}, \mathbb{Q}, \mathbb{A}*$ ma non è possibile utilizzare $\mathbb{R}$ perché non è numerabile.

Una volta fissato un dominio con una codifica effettiva, una funzione $f : D \to D$ è **calcolabile** se $f^* = \alpha \circ f \circ \alpha^{-1}$ è URM-Calcolabile.

Ad esempio: $Q = \mathbb{Z}$, $f : \mathbb{Z} \to \mathbb{Z}$, $f(z) = |z|$ è calcolabile perché è possibile trovare una funzione $\alpha : \mathbb{Z} \to \mathbb{N}$ che codifica i numeri positivi con i numeri pari e quelli negativi con i numeri dispari.

Così facendo viene stabilita una corrispondenza *1-1* tra $\mathbb{Z}$ e $\mathbb{N}$.

$$\alpha(z) = \begin{cases} 2z & \text{se } z \text{ è positivo} \\ -2z - 1 & \text{se z è negativo} \end{cases}$$

L'inversa viene definita come

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

Pertanto f^* risulta essere

$$f^* = \begin{cases} \alpha(f(\frac{n}{2})), & \text{se } n \text{ è pari} \\ \alpha(f(\frac{-(n+1)}{2})), & \text{se } n \text{ è dispari} \end{cases}$$

$$f^* = \begin{cases} n, & \text{se } n \text{ è pari} \\ n+1, & \text{se } n \text{ è dispari} \end{cases}$$

Capitolo 2

Composizione di funzioni calcolabili

Si possono ottenere delle funzioni calcolabili utilizzando delle composizioni di funzioni calcolabili più semplici.

La combinazione può essere fatta mediante:

- **Composizione funzionale**
- **Ricorsione primitiva**, ovvero che termina sempre
- **Minimizzazione illimitata**, ovvero l'iterazione fino a quando una certa condizione non viene soddisfatta.

In termini più tecnici, la classe $\mathcal{C}$ è **chiusa** rispetto alle operazioni sopra riportate.

Ci si può spingere oltre, dimostrando che tutte le funzioni calcolabili possono essere ottenute da combinazioni di funzioni calcolabili elementari.

Funzioni di base: insieme delle funzioni calcolabili elementari:

- $\underline{0} : \mathbb{N} \to \mathbb{N}^k$, $\underline{0}(x_1, \ldots, x_k) = 0$
- $succ : \mathbb{N} \to \mathbb{N}$, $succ(x) = x + 1$
- $U_i^k : \mathbb{N}^k \to \mathbb{N}$, $U_i^k(x_1, \ldots, x_k) = x_i$

e possono essere tutte calcolate da delle singole istruzioni URM.

2.1 Notazioni e ipotesi per i programmi

Per dimostrare la composizione di funzioni calcolabili viene fatta una composizione dei programmi che le calcolano.

Per rendere le varie dimostrazioni più semplici si utilizzando varie notazioni/ipotesi:

- Dato un programma P in URM, $\rho(P) = max(\{i | R_i \text{ è usato in } P\})$
- P è in forma standard, ovvero tutti i salti che fanno terminare il programma vanno a finire all'istruzione *L(P) + 1*.
- La concatenazione dei programmi viene indicata con $P\ Q$ e si assume che questa sia corretta rispetto ai salti a fine programma, ovvero che quando viene effettuata la concatenazione, le istruzioni di salto vengano aggiornate in modo opportuno.
- Dato P, $P[i_1, \ldots, i_k \to l]$ indica il programma che prende in input i registri $i_1 \ldots, i_k$ e mette l'output in l e non assume niente sugli altri registri, ovvero modifica P in

```
T(i1,1)
...
T(ik,k)
```

```
Z(k+1)
...
Z(rho(P))
P
T(1,l)
```

C'è un piccolo problema con i vari trasferimenti `T(ij,j)` che possono sovrascrivere alcuni dati, pertanto è necessario stare attenti che questo non succeda utilizzando dei registri di supporto.

2.2 Composizione generalizzata

$$f : \mathbb{N}^k \to \mathbb{N},\ g_1, \dots g_n : \mathbb{N}^k \to \mathbb{N}$$

La loro composizione $h : \mathbb{N}^k \to \mathbb{N}$ è data da:

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

La funzione $h(\vec{x}) \downarrow$ (è definita) se tutte le $g_i \downarrow y_i, f(y_1, \dots, y_n) \downarrow$.

L'approccio utilizzato nella valutazione delle funzioni è quello **eager**, ovvero vengono valutati prima tutti i parametri.

Ad esempio: $\underline{0} : \mathbb{N} \to \mathbb{N}$, $\underline{0}(x) = 0$ e $d(x) = \uparrow$, $\underline{0}(d(1))$ è $\uparrow$ perché prima è necessario valutare i parametri.

2.2.1 Calcolabilità della funzione composta

Se $f : \mathbb{N}^n \to \mathbb{N}, g_1 \dots g_n : \mathbb{N}^k \to \mathbb{N} \in \mathcal{C}$, allora anche $h : \mathbb{N}^k \to \mathbb{N}$ è calcolabile in $\mathcal{C}$.

Dimostrazione

Siano $F, G_1, \dots, G_n$ programmi URM in forma normale per le relative funzioni.

L'input della funzione h avrà nei primi k registri i valori di input, è necessario quindi andare a copiarli in una locazione di memoria che non viene usata dai vari programmi, ovvero dalla locazione $m+1$, con $m = max\{\rho(F), \rho(G_1), \dots \rho(G_N), k, n\}$.

I risultati parziali dei programmi vengono poi memorizzati a partire dalla locazione $m + k + 1$ per poi essere utilizzati da F

Il programma risultante è:

```
T([1 ... k],[m+1 ... m+k])
G1[m+1 ... m+k -> m+k+1]
...
Gn[m+1 ... m+k -> m+k+n]
F[m+k+1 ... m+k+n -> 1]
```

Esempio - Somma di due numeri

A partire dalla funzione $sum(x_1, x_2) = x_1 + x_2$ è possibile andare a ottenere la funzione

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

componendo la funzione sum con se stessa:

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

Tuttavia, strettamente parlando, le g_i non hanno la stessa arietà, pertanto sono necessari dei piccoli aggiustamenti:

$$f(x_1, x_2, x_3) = sum(sum(U_1^3(\vec{x}), U_2^3(\vec{x}))), U_3^3(\vec{x}))$$

2.3 Ricorsione Primitiva

$$\begin{aligned} fact(0) &= 1 \\ fact(n+1) &= (n+1)fact(n) \end{aligned}$$

$$\begin{aligned} fib(0) &= 1 \\ fib(1) &= 1 \\ fib(n+2) &= fib(n+1) + fib(n) \end{aligned}$$

Date $f : \mathbb{N}^k \to \mathbb{N}$ e $g : \mathbb{N}^{k+2} \to \mathbb{N}$, la funzione per ricorsione primitiva $h : \mathbb{N}^{k+1} \to \mathbb{N}$ è definita come

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

Così facendo viene definita h utilizzando h e concettualmente è corretto, tuttavia è necessario dimostrare formalmente l'esistenza e l'unicità di h.

Questo lo si fa considerando l'operatore Φ:

$$\begin{aligned} \Phi : (\mathbb{N}^k \to \mathbb{N}) &\to (\mathbb{N}^k \to \mathbb{N}) \\ \Phi(h)(\vec{x}, 0) &= f(\vec{x}) \\ \Phi(h)(\vec{x}, y+1) &= g(\vec{x}, y, h(\vec{x}, y)) \end{aligned}$$

tale che $\Phi(h) = h$, ovvero tale che viene raggiunto un punto fisso. Ciò sarebbe da dimostrare, ma questo va oltre l'obiettivo del corso, quindi viene dato per buono.

In altre parole, se h rispetta lo schema precedentemente definito, allora esiste ed è unica.

Assumendo quindi che la ricorsione primitiva di una funzione calcolabili è calcolabile, è possibile definire varie operazioni senza scrivere esplicitamente il programma per calcolarle:

Ad esempio la funzione somma può essere definita in modo ricorsivo:

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

In modo simile possono essere anche definiti il prodotto e l'esponenziale:

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

$$x^y = h(x, y) = \begin{cases} x^0 = 1 & \Rightarrow f(x) = 1 \\ x^{(y+1)} = (x^y) \cdot x & \Rightarrow g(x, y, z) = z \cdot x \end{cases}$$

2.3.1 Calcolabilità della Ricorsione Primitiva

Siano $f : \mathbb{N}^k \to \mathbb{N}$ e $g : \mathbb{N}^{k+2} \to \mathbb{N} \in \mathcal{C}$ allora anche h definita per ricorsione primitiva è in $\mathcal{C}$. Ovvero quello che è stato assunto precedentemente.

Dimostrazione

Siano F e G i programmi che calcolano f e g.

Il programma che calcola h verrà invocato con il vettore x nelle prime k locazioni e con y nella locazione $k+1$.

Come prima cosa è necessario trasferire i dati di input in una zona di memoria non utilizzata dai programmi F e G, ovvero $m+1$, dove $m = max\{\rho(F), \rho(G), k+2\}$.

Dopodiché è necessario effettuare un'interazione su un contatore i che parte da 0, fino a quando non viene raggiunto y.

```
h(vec(x), 0) = f(vec(x)) -- i=0 -- i==y? NO
h(vec(x), 1) = g(vec(x), 0, h(vec(x),0)) -- i=1 -- i==y? NO
...
h(vec(x), i+1) = g(vec(x), i, h(vec(x),i)) -- i=n -- i==y? SI -> fine
```

ovvero il programma per h sarà:

```
1 T([1..k], [m+1 ... m+k]) // copia x
2 T(k+1, m+k+3) //copia y
3 F[m+1 ... m+k -> m+k+2]
4 J(m+k+3, m+k+1, END) #LOOP
5 G[m+1 ... m+k+2 -> m+k+2] // h(vec(x),i+1)
6 S(m+k+1) //i++
7 J(1,1,LOOP)
8 T(m+k+2,1) #END
```

Se f e g sono funzioni parziali quanto dimostrato richiede maggiori precisazioni, ma a noi basta sapere che se durante il calcolo troviamo qualche funzione non definita, anche h non è definita.

2.3.2 Funzioni totali definite ricorsivamente

Le funzioni definite a parte da funzioni totali mediante composizione o ricorsione sono anche loro totali.

La dimostrazione per le funzioni mediante composizione è vera per definizione, l'altra è lasciata per esercizio (si fa per induzione).

todo

2.3.3 Esercizio - Libreria di funzioni calcolabili

- somma $x+y$
- prodotto $x \cdot y$
- esponenziale x^y
- fattoriale $fact(x)$

Sguardo d'insieme: vogliamo trovare un programma URM in grado di simulare un altro programma URM, le operazioni numeriche sono interessanti perché il programma in input verrà rappresentato come un numero.

Predecessore

$$\text{pred}(x) = \begin{cases} x \dot{-} 1 = 0, & \text{se } x = 0 \\ x - 1, & \text{se } x > 0 \end{cases}$$

Può essere calcolata ricorsivamente con

$$0 \dot{-} 1 = 0$$
$$(y+1) \dot{-} 1 = y$$

Sottrazione tra numeri naturali

$$x \dot{-} y = \begin{cases} 0, & \text{se } x \leq y \\ x - y, & \text{altrimenti} \end{cases}$$

Può essere calcolata ricorsivamente con:

$$x \dot{-} 0 = x$$
$$x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1$$

Segno

$$sg(x) = \begin{cases} 0, & \text{se } x = 0 \\ 1, & \text{altrimenti} \end{cases}$$

Può essere calcolata ricorsivamente con:

$$sg(0) = 0$$
$$sg(y+1) = 1$$

In modo simile può essere definito anche $\overline{sg}$.

Valore assoluto della differenza

$$|x - y| = \begin{cases} x - y, & \text{se } x \leq y \\ y - x, & \text{altrimenti} \end{cases}$$

Può essere calcolata in modo composizionale con:

$$|x - y| = (x \dot{-} y) + (y \dot{-} x)$$

Minimo

$$\min(x, y) = \begin{cases} x, & \text{se } x \leq y \\ y, & \text{altrimenti} \end{cases}$$

Può essere calcolata con:

$$\min(x, y) = x \dot{-} (x \dot{-} y)$$

Questo perché se x è il minimo, $x \dot{-} y$ è 0.

Resto della divisione intera

$$\text{rm}(x,y) = \begin{cases} y \mod x, & \text{se } x \neq y \\ y, & \text{altrimenti} \end{cases}$$

Può essere calcolato ricorsivamente come:

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

L'*if* può essere espresso in modo algebrico utilizzando la funzione *sg*, ottenendo:

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

Esercizio - Divisione intera

$$\text{qt}(x,y) = \begin{cases} \left\lfloor \frac{y}{x} \right\rfloor, & \text{se } x \neq y \\ y, & \text{altrimenti} \end{cases}$$

todo

Esercizio - Definizione per casi

$f_1 \dots f_n : \mathbb{N}^k \to \mathbb{N}$ totali e calcolabili e $Q_1, \dots, Q_n \subseteq \mathbb{N}^k$ decidibili e tali che per ogni $\vec{x}$ solo un predicato è vero.

Definire $f(x) = f_1(x)$ se $Q_1(x) \dots f_n(x)$ se $Q_n(x)$

todo

Soluzione La funzione *f(x)* viene definita per casi e dal momento che i vari predicati sono decidibili ed esaustivi, quindi almeno uno dei casi è vero.

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

Quando un caso è vero, viene calcolata la funzione associata, che è totale. Se questa funzione non fosse totale, potrebbe essere che il predicato vero su un certo x, ma che la funzione su quell'x non sia definita e quindi neanche *f(x)* risulterebbe definita, perdendo così la totalità.

La calcolabilità deriva dal fatto che la definizione per casi viene fatta con una serie di *if* che è dimostrato essere calcolabile e le funzioni $\mathcal{X}_{Q_i}$ sono calcolabili, perché i predicati sono decidibili.

Algebra della decidibilità

I predicati decidibili sono chiusi rispetto negazione, congiunzione e disgiunzione.

Ovvero se Q e Q' sono due predicati in $\mathbb{N}^k$ decidibili allora anche:

1. $\neg Q(\vec{(x)})$
2. $Q(\vec{x}) \wedge Q'(\vec{x})$
3. $Q(\vec{x}) \vee Q'(\vec{x})$

Questo perché le relative funzioni che li calcolano possono essere definite come:

1. $\mathcal{X}_{\neg Q}(\vec{x}) = \overline{sg}(\mathcal{X}_Q(\vec{x}))$

2. $\mathcal{X}_{Q \wedge Q'}(\vec{x}) = \mathcal{X}_Q(\vec{x}) \cdot \mathcal{X}_{Q'}(\vec{x})$
3. $\mathcal{X}_{Q \vee Q'}(\vec{x}) = sg(\mathcal{X}_Q(\vec{x}) + \mathcal{X}_{Q'}(\vec{x}))$

Tutte le funzioni così definite sono calcolabili perché ottenute da composizioni di funzioni calcolabili.

Somma e prodotto dei valori di una funzione

Data una funzione totale e $f : \mathbb{N}^{k+1} \to \mathbb{N}$ totale e calcolabile, è possibile definire le due funzioni che effettuano la somma e il prodotto dei primi y valori della funzione.

$$s(\vec{x}, y) = \sum_{z<y} f(\vec{x}, z)$$
$$p(\vec{x}, y) = \prod_{z<y} f(\vec{x}, z)$$

Entrambe le funzioni sono calcolabili e totali perché possono essere definite induttivamente (ricorsivamente) a partire da delle funzioni calcolabili.

$$s(\vec{x}, y) = \begin{cases} \sum_{z<0} f(\vec{x}, z) = 0, & \text{se } y = 0 \\ \sum_{z<y+1} f(\vec{x}, z) = \sum_{z<y} f(\vec{x}, z) + f(\vec{x}, z), & \text{altrimenti} \end{cases}$$
$$p(\vec{x}, y) = \begin{cases} \prod_{z<0} f(\vec{x}, z) = 1, & \text{se } y = 0 \\ \prod_{z<y+1} f(\vec{x}, z) = \prod_{z<y} f(\vec{x}, z) \cdot f(\vec{x}, z) & \text{altrimenti} \end{cases}$$

La totalità deriva dal fatto che f è totale.

Quantificazione limitata

Combinando algebra della decidibilità e quanto detto nel paragrafo precedente è possibile la decidibilità di $\forall$ e $\exists$.

Dato un predicato $Q(\vec{x}, z)$ per calcolare se $\forall z < y, Q(\vec{x}, z)$ è possibile utilizzare la funzione:

$$\mathcal{X}_{Q_\forall} = \prod_{z<y} \mathcal{X}_Q(\vec{x}, z)$$

In modo simile è possibile calcolare $\exists z < y, Q(\vec{x}, z)$:

$$\mathcal{X}_{Q_\exists} = sg(\sum_{z<y} \mathcal{X}_Q(\vec{x}, z))$$

Trattandosi della composizione di funzioni calcolabili e totali, le funzioni così ottenute sono a loro volta calcolabili e totali.

2.4 Minimalizzazione limitata

Data una funzione $f(\vec{x}, z) : \mathbb{N}^{k+1} \to \mathbb{N}$ calcolabile e totale è possibile definire una funzione

$$h(\vec{x}, y) = \mu z < y | f(\vec{x}, z) = 0$$

h è ancora una funzione $\mathbb{N}^{k+1} \to \mathbb{N}$ e viene calcolata come il minimo valore di z minore di y e tale che $f(\vec{x}, z)$ sia uguale a 0 (tipicamente l'uguale a 0 viene omesso).

La definizione più precisa è:

$$h(\vec{x}, y) = \mu z < y.f(\vec{x}, z) = \begin{cases} \text{minimo } z < y \text{ tale che } f(\vec{x}, z) = 0 \text{ se questo esiste} \\ y, \text{altrimenti} \end{cases}$$

Per come è definita, questa funzione risulta essere **totale** e **calcolabile**. Intuitivamente è calcolabile perché si tratta di calcolare f per vari valori, serve però una dimostrazione più formale, fatta per ricorsione primitiva.

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

$$h(\vec{x}, y+1) = \begin{cases} h(\vec{x}, y) < y, & f \text{ si annulla su un valore minore di } y, \text{ viene resituito } h(\vec{x}, y) \\ h(\vec{x}, y) = y, & \text{per tutti i valori di minori } y \text{ non c'è uno } 0 \begin{cases} \text{se } f(\vec{x}, y) = 0 \to y \\ \text{se } f(\vec{x}, y) \neq 0 \to y+1 \end{cases} \end{cases}$$

La seconda parte può essere facilmente tradotta nell'espressione

$$(y \dot{-} h(\vec{x}, y)) \cdot (h(\vec{x}, y)) + \overline{sg}(y \dot{-} h(\vec{x}, y)) \cdot (y + sg(f(\vec{x}, y)))$$

Dal momento che la funzione h può essere definita per ricorsione primitiva e per composizione di funzioni calcolabili, anche lei è calcolabile.

2.4.1 Funzioni calcolabili per ricorsione limitata

Utilizzando la ricorsione limitata è possibile dimostrare la calcolabilità di varie funzioni.

Numero di divisori di x

$$\begin{aligned} D(x) &= \#\text{ divisori di } x \\ &= \sum_{y \leq x} (\overline{sg}(rm(y, x))) \end{aligned}$$

Dove rm è la funzione resto, precedentemente dimostrata calcolabile.

Numeri primi

Dimostrare la calcolabilità dei funzioni che lavorano con i numeri primi è importante perché torneranno utili in futuro.

$$\begin{aligned} Pr(x) &= \text{“}x \text{ è primo ”} \\ &= \overline{sg}(|D(x) - 2|) \end{aligned}$$

$$\begin{aligned} P_x &= \text{“}x\text{-esimo numero primo, per convezione: ”} P_0 = 0, P_1 = 2, \ldots \\ &= \begin{cases} P_0 = 0 \\ P_{x+1} = \mu z \leq (P_x! + 1) \, |Pr(z) \cdot \underbrace{\overline{sg}(P_x + 1 \dot{-} z)}_{1 \text{ se } z > P_x} \dot{-} 1| \end{cases} \end{aligned}$$

$$\begin{aligned} (x)_y &= \text{esponente di } P_y \text{ nella decomposizione di } y \\ &= \max z \, P_y^z \text{ divide } x \to \mu z \leq x \text{ tale che } P_y^{z+1} \text{ non divide } x \\ &= \mu z \leq x \, \overline{sg}(rm(P_y^{z+1}, x)) \end{aligned}$$

Esercizio - mcm, MCD, radice di x

Dimostrare che sono calcolabili:

- $floor(\sqrt{x})$
- $mcm(x, y))$
- $MCD(x, y)$

2.5 Codifica di coppie

La funzione di *fibonacci* non può essere definita per ricorsione primitiva, perché il passo induttivo richiede una coppia di valori precedenti.

È però possibile definire una funzione $\prod : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ che codifica il valore di una coppia in un unico numero:

$$\prod(x, y) = 2^x(2y + 1) \dot{-} 1$$

Questa funzione risulta essere biunivoca perché è possibile definire l'inversa:

$$\prod^{-1}(x) = ((n + 1)_1, \frac{1}{2}(\frac{n + 1}{(n + 1)_1}) - 1)$$

La funzione inversa è effettiva, perché è definita in termini di componenti calcolabili, anche se la definizione di funzione calcolabile non è stata vista per funzioni $\mathbb{N} \to \mathbb{N} \times \mathbb{N}$.

Utilizzando la funzione accoppiamento, è possibile definire la funzione *fib* per ricorsione primitiva:

$$\begin{aligned}
g(x) &= \prod(fib(x), fib(x + 1)) \\
g(0) &= \prod(fib(0), fib(1)) = \prod(1, 1) = 5 \\
g(x + 1) &= \prod(fib(x + 1), fib(x + 2)) \\
&= \prod(fib(x + 1, fib(x) + fib(x + 1)) \\
&= \prod(\prod_2(g(x)), \prod(\prod_1(g(x)) + \prod_2(g(x)))
\end{aligned}$$

Dove $\prod_1$ e $\prod_2$ sono rispettivamente le funzioni per il calcolo del primo e del secondo elemento della coppia.

Così facendo è stata dimostrata la calcolabilità di g per ricorsione primitiva, ma $fib(x) = \prod_1(g(x))$ e di conseguenza *fib* è calcolabile per composizione di funzioni calcolabili.

2.6 Minimalizzazione illimtata

Data $f : \mathbb{N}^{k+1} \to \mathbb{N}$, si vuole definire $h : \mathbb{N}^k \to \mathbb{N}$ tale che calcoli il minimo z che azzera la funzione f, ovvero:

$$h(\vec{x}) = \mu z.f(\vec{x}, z)$$

Ci sono però dei problemi se $f(\vec{x}, z)$ è sempre diversa da zero, perché in questo caso h è $\uparrow$.

Un altro problema si ha se la funzione è indefinita per un valore z' minore dello z che azzera la funzione. Anche in questo caso si ha che h è $\uparrow$.

$$h(\vec{x}) = \mu z.f(\vec{x}, z) = \begin{cases} \text{minimo } z \text{ tale che } f(\vec{x}, z) = 0 \text{ se esiste e se } \forall z' < z, f(\vec{x}, z') \downarrow \neq 0 \\ \uparrow \text{ altrimenti} \end{cases}$$

Alternativamente, definendo $Z_{f,\vec{x}} = \{z | f(\vec{x}, z) = 0 \wedge \forall z' < z f(\vec{x}, z') \downarrow\}$, si ha che h è definita come

$$h(\vec{x}) = \begin{cases} \min Z_{f,\vec{x}} \text{ se } Z_{f,\vec{x}} \neq \emptyset \\ \uparrow \text{ altrimenti} \end{cases}$$

2.6.1 Esercizi

Esercizio - Radice quadrata

Dimostrare la calcolabilità di

$$f(x) = \begin{cases} \sqrt{x} \text{ se } x \text{ è un quadrato} \\ \uparrow \text{ altrimenti} \end{cases}$$

Soluzione L'idea è quella di trovare un y che elevato al quadrato è uguale a x: $y^2 - x = 0$.

Si ha quindi che

$$f(x) = \mu y.|y^2 - x|$$

ed è calcolabile perché minimizza illimitatamente una composizione di funzione calcolabile.

Esercizio teorico

Dimostrare che se $f : \mathbb{N} \to \mathbb{N}$ è iniettiva, calcolabile e totale, anche la sua inversa è calcolabile.

Soluzione

$$f^{-1}(x) = \begin{cases} y, & \text{tale che } f(y) = x \\ \uparrow, & \text{altrimenti} \end{cases}$$

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

Perché l'uguaglianza sia rispettata è necessario che f sia totale, dal momento che se per un certo y, f non è definita il programma che la calcola non termina, rendendo indefinita anche f^{-1}.

C'è un barbatrucco per gestire anche la non totalità di f, ovvero quello di eseguire contemporaneamente il calcolo per ogni y, eseguendo tot passi alla volta per ognuno dei calcoli (*dovrebbe essere dimostrato in futuro*).

Dal momento che f è calcolabile si ha che anche l'inversa è calcolabile per minimizzazione illimitata di una composizione di funzioni calcolabili.

L'iniettività[1] garantisce che il valore trovato sia quello corretto.

Esercizio - Divisione

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

[1] Una funzione $f : X \to Y$ si dice iniettiva se due elementi distinti del dominio hanno immagini distinte, ovvero $a_1 \neq a_2$ implica $f(a_1) \neq f(a_2)$.

Soluzione

$$f(x,y) = \mu k.|x \dot{-} y \cdot k|$$

C'è però un problema, perché la funzione così definita risulta calcolabile se x e y sono uguali a 0.

$$f(x,y) = \mu k.(|x - y \cdot k| + \underbrace{\overline{sg}(y)}_{\text{vale 1 se } y \text{ è uguale a 0}})$$

Così facendo se $y = 0$ la minimalizzazione non converge.

Questo porta ad un discorso un po' più ampio sulla possibilità di aggiustare una funzione che si comporta quasi come un'altra funzione (§**??**).

Calcolabilità della minimalizzazione

Se $f : \mathbb{N}^{k+1} \to \mathbb{N}$ è in $\mathcal{C}$, allora anche $\mu y.f(\vec{x}, y)$ è in $\mathcal{C}$.

Dimostrazione Sia $f : \mathbb{N}^{k+1} \to \mathbb{N}$ in $\mathcal{C}$ e sia P il programma in forma normale che calcola f.

L'idea è quella di eseguire P incrementando via via un contatore e, quando viene trovato un valore del contatore che azzera la funzione, questo viene ritornato.

```
 1       k          m+1           m+k|m+k+1|m+k+2|
|\vec{x} | .....  |x_1|x_2|....|x_k|  0   |  0   |
```

Sia $m = \max \rho(P), k$, il programma che calcola la minimalizzazione illimitata è:

```
1 T([1..k], [m+1..m+k]) //Copio l'input
2 P[m+1, ..., m+k+1 -> 1]
3 J(1, m+k+2, END) #LOOP
4 S(m+k+1)
5 J(1,1,LOOP)
6 #END
```

Dal momento che è possibile trovare un programma che calcola la minimalizzazione illimitata, questa è calcolabile.

Hot-fix delle funzioni

Data una funzione $f : \mathbb{N} \to \mathbb{N}$ tale che esiste $g : \mathbb{N} \to \mathbb{N}$ calcolabile e tale che

$$\Delta = \{x | f(x) \neq g(x)\}$$

e finito.

Allora f è calcolabile e può essere modificata in modo da coincidere con g.

Funzioni finite

Funzione finita $\Theta : \mathbb{N}^k \to \mathbb{N}$è una funzione finita quando è definita come:

$$\Theta(\vec{x}) = \begin{cases} y_1, & \text{se } \vec{x} = \vec{x}_1 \\ y_2, & \text{se } \vec{x} = \vec{x}_2 \\ \cdots & \\ y_n, & \text{se } \vec{x} = \vec{x}_n \\ \uparrow, & \text{altrimenti} \end{cases}$$

Tutte le funzioni di questo tipo sono calcolabili

Dimostrazione (per semplicità è ridotta a funzione unarie)

$$\Theta = \begin{cases} y_1, & \text{se } x = x_1 \\ y_2, & \text{se } x = x_2 \\ \dots & \\ y_n, & \text{se } x = x_n \\ \uparrow, & \text{altrimenti} \end{cases}$$

$$= y_1 \cdot \overline{sg}(|x - x_1|) + \dots + y_n \cdot \overline{sg}(|x - x_n|) + \underbrace{\mu z. \prod_{i=1}^{n} |x - x_i|}_{\text{funzione indipendente da } z}$$

La minimalizzazione su z è calcolabile ma risulta indefinita, perché non è possibile minimizzarla rispetto a z, quindi anche la funzione Θ risulta essere indefinita se tutti i valori sono diversi da $x_1 \dots x_n$.

2.7 Tesi di Church

Ogni funzione è calcolabile tramite un procedimento effettivo se e solo se è URM calcolabile.

Church non ha proprio detto questo, perché non c'era URM quando è stata enunciata e ha utilizzato un modello alternativo: le funzioni parziali ricorsive $\mathcal{R}$ (Göedel).

La classe $\mathcal{R}$ delle **funzioni parziali ricorsive** è la **minima** classe di funzioni che contiene:

(a) zero: $z(\vec{x}) = 0$ per ogni x

(b) successore: $S(x) = x + 1$

(c) proiezioni: $U_i^k(x_1 \dots x_k) = x_i$

ed è chiusa rispetto:

1. composizione generalizzata
2. ricorsione primitiva
3. minimalizzazione illimitata.

Una classe di funzioni X è **ricca** se contiene le funzioni di base ed è chiusa rispetto alle 3 operazioni classiche.

C'è almeno una classe ricca, perché anche la classe "tutte le funzioni" è ricca, anche se ha poco senso considerarla.

Si vuole quindi che $\mathcal{R}$ sia contenuta in X ricca e che anche $\mathcal{R}$ sia ricca.

Questo è possibile perché l'intersezione di due classi ricche è ovviamente anch'essa ricca.

$\mathcal{R}$ può essere definita come l'intersezione di tutte le classi di funzioni ricche.

$$\mathcal{R} = \bigcap_{X \text{ ricca}} X$$

anche se precedentemente $\mathcal{R}$ è stata definita come

$$\mathcal{R} = \{(a), (b), (c) \text{ con tutte le funzioni che si ottengono da queste utilizzando } 1, 2, 3\}$$

È dimostrabile che le due definizioni sono equivalenti, anche se non lo dimostriamo.

Si può anche definire la classe $\mathcal{PR}$ delle **funzioni primitive ricorsive**: la minima classe che contiene solamente le funzioni di base chiusa rispetto la composizione e la ricorsione primitiva. Ovviamente $\mathcal{PR}$ è più piccola di $\mathcal{R}$.

2.7.1 $\mathcal{C} = \mathcal{R}$

È già stato dimostrato che $\mathcal{C}$ è una classe ricca, quindi sicuramente $\mathcal{R} \subseteq \mathcal{C}$.

Resta da dimostrare che $\mathcal{C} \subseteq \mathcal{R}$.

Sia f in $\mathcal{C}$, ovvero esiste un programma P in forma normale che la calcola $f = f_p^{(k)}$.

```
P= I_1 ... I_s

|x_1 ... x_k| 0 ... 0
```

Supponiamo di avere le funzioni:

$$C_p^1(\vec{x}, y) = \begin{cases} \text{contenuto di R1 dopo } t \text{ passi del programma se non è terminato} \\ \text{altrimenti ritorna il valore finale del registro} \end{cases}$$

$$J_p(\vec{x}, t) = \begin{cases} \text{la prossima istruzione da eseguire all'istante } t \text{ di } P(\vec{x}) \text{ se non è terminato} \\ 0 \text{ altrimenti} \end{cases}$$

Si ha che entrambe le funzioni sono del tipo $\mathbb{N}^{k+1} \to \mathbb{N}$ e totali.

Se $f(x) \downarrow$ allora P termina su $\vec{x}$ in un qualche numero di passi

$$t_0 = \mu t.J_p(\vec{x}, t)$$

e

$$f(\vec{x}) = C_p^1(\vec{x}, t_0) = C_p^1(\vec{x}, \mu t.J_p(\vec{x}, t))$$

Se invece $f(\vec{x}) \uparrow$ si ha che

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

e

$$f(\vec{x}) = C_p^1(\vec{x}, t_0) = C_p^1(\vec{x}, \mu t.J_p(\vec{x}, t))$$

Ovvero la combinazione di queste funzioni riesce a descrivere un programma URM.

Resta da dimostrare che queste due funzioni sono contenute in $\mathcal{R}$ e pertanto, dato che f è la combinazione di queste funzioni, anche f è in $\mathcal{R}$.

Il programma P sarà fatto da un certo numero di istruzioni che andranno a modificare determinati registri.

L'idea è quella di andare a codificare la configurazione di memoria $< r_1 r_2 \ldots . >$ utilizzata dal programma con un numero

$$x = \prod_{i \geq 1} p_i^{r_i}$$

Da notare che il prodotto non è infinito, perché una volta fissato il programma, il numero di registri che questo utilizza è limitato.

Per come è definita la codifica si ha che

$$r_i = (x)_i (\text{esponente dell'} i\text{-esimo numero primo nella decoposizione del numero } x)$$

È quindi possibile definire una funzione $C_p : \mathbb{N}^{k+1} \to \mathbb{N}$, $C_p(\vec{x}, t) =$ contenuto della memoria dopo t passi e $J_p : \mathbb{N}^{k+1} \to \mathbb{N}$ con lo stesso significato di prima.

Le due funzioni possono essere raccolte nella coppia

$$\vartheta_p = \Pi(C_p(\vec{x}, t), J_p(\vec{x}, t))$$

che risulta definita per ricorsione primitiva con

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

Abbreviando con $C = C_p(\vec{x}, t)$ e $J = J_p(\vec{x}, t)$, si può definire il caso ricorsivo come

$$C_p(\vec{x}, t+1) = \begin{cases} qt(p_n^{(C)_n}) & \text{se } 1 \leq J \leq S \text{ e } I_J = Z(n) \\ p_n \cdot C & \text{se } 1 \leq J \leq S \text{ e } I_J = S(n) \\ p_n^{(C)_m} \cdot qt(p_n^{(C)_n}) & \text{se } 1 \leq J \leq S \text{ e } I_J = T(m,n) \\ C & \text{altrimenti} \end{cases}$$

Nel caso 1 viene fatto il quoziente del numero da azzerare in modo da azzerare l'esponente relativo al registro nella rappresentazione numerica della memoria.

Nel caso 2 viene fatta la moltiplicazione del numero primo associato al registro per aumentare di 1 l'esponente del numero primo associato al registro nella rappresentazione numerica della memoria.

Il caso altrimenti si verifica quando il programma termina oppure l'istruzione di salto punta fuori dal programma, in entrambi i casi la memoria del programma non cambia.

$$J_p(\vec{x}, t+1) \begin{cases} J+1 & \text{se } 1 \leq J < S \text{ e } (I_J = Z(n), S(n), T(m,n)) \text{ oppure } (I_J = J(m,n,q), (C)_m \neq (C)_n) \\ q & \text{se } 1 \leq J < S \text{ e } I_J = J(m,n,q) \text{ e } (C)_m = (C)_n \text{ e } 1 \leq q < S \\ 0 & \text{altrimenti} \end{cases}$$

Nel caso 1, se l'istruzione non è salto, oppure è un salto ma con condizione falsa, l'istruzione *t + 1* è quella successiva. Se il salto ha condizione è vera e l'istruzione a cui saltare è interna al programma, l'istruzione successiva è *q*. Altrimenti se il programma è terminato oppure l'istruzione di salto è finita fuori dal programma viene ritornato 0.

Le due funzioni vengono quindi definite utilizzando una definizione "per casi" e combinando delle funzioni definite precedentemente con le operazioni di base (primitive ricorsive), quindi anche queste due funzioni sono in $\mathcal{PR}$ e quindi sono anche in $\mathcal{R}$.

Quindi anche

$$f(\vec{x}) = C_P(\vec{x}, \mu t. J_P(\vec{x}, t)) \in \mathcal{R}$$

2.8 Funzioni primitive ricorsive

$\mathcal{PR}$ è la minima classe di funzioni che contiene le funzioni:

(a) zero

(b) succ

(c) proiezione

e chiusa rispetto a:

1. composizione
2. ricorsione primitiva

Si ha che $URM_{for,while}$ definisce $\mathcal{C}_{for,while} = \mathcal{C} = \mathcal{R}$.

Se non viene utilizzato il `while` $\mathcal{C}_{for} = \mathcal{PR} \subsetneq \mathcal{R} = \mathcal{C}$

Questo perché senza il `while` possono essere calcolate solo le funzioni totali.

Uno si può chiedere se il `while` serve solo a fare confusione o se serve anche per calcolare funzioni totali. Ovvero

$$\mathcal{PR} = \mathcal{R} \cap Tot \text{ oppure } \mathcal{PR} \subsetneq \mathcal{R} \cap Tot$$

Questo si dimostra utilizzando la **funzione di Ackermann**:

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

Gli argomenti delle chiamate ricorsive di questa funzione diminuisce secondo l'ordine lessicografico.

$$(x, y) \leq_{lex} (x', y') \begin{cases} \text{se } x < x' \\ \text{oppure } x = x' \text{ e } y < y' \end{cases}$$

Una caratteristica di questo ordine è che è **ben fondato**, ovvero non ha catene discendenti infinite.

Più formalmente $(D, \leq)$ è ben fondato se non ha sequenze del tipo $d_0 > d_1 > d_2 > \ldots$.

Se la relazione d'ordine è ben fondata e ad ogni passo viene prodotta una configurazione strettamente maggiore, si ha la garanzia che la sequenza di invocazioni termini.

$$\underbrace{(0,0)(0,1)(0,2)\ldots}_{\mathbb{N}}\underbrace{(1,0)(1,1)(1,2)\ldots}_{\mathbb{N}}\underbrace{(2,0)(2,1)(2,2)\ldots}_{\mathbb{N}}$$

Ogni sottoinsieme non vuoto di $(\mathbb{N}^2, \leq_{\{} lex\})$ ha un minimo dato $X \subseteq \mathbb{N}^2$.

$$\begin{aligned} x_0 &= \min\{x \mid (x, y) \in X\} \\ y_0 &= \min\{y \mid (x_0, y) \in X\} \end{aligned}$$

Tutto questo per dire che la funzione di Ackermann ha una ricorsione sensata.

Induzione ben fondata: Dato $(D, \leq)$ è ben fondato e P è una qualche proprietà.

Se $(\forall d \in D(\forall d' < dP(d')) \Rightarrow P(d)) \Rightarrow \forall d \in DP(d)$. Questo ci serve per dimostrare le proprietà della funzione di Ackermann.

2.8.1 Totalità della funzione di Ackermann

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

Dimostrazione

Sia $(x, y) \in \mathbb{N}^2$ tale che $\forall (x', y') <_{lex} (x, y)$, $\psi(x', y') \downarrow$

allora $\psi(x, y) \downarrow$

Ci sono vari casi

$$\begin{aligned} (x = 0)\psi(x, y) &= \psi(0, y) = y + 1 = \downarrow \\ (x > 0, y = 0)\psi(x, y) &= \psi(x, 0) = \underbrace{\psi(x - 1, 1)}_{<_{lex}} = \downarrow \text{ per ipotesi induttiva} \\ (x > 0, y > 0)\psi(x, y) &= \psi(x - 1, \underbrace{\psi(x, y - 1)}_{\text{per ipotesi induttiva è } \downarrow = u}) = \psi(x - 1, u) = \downarrow \text{ per ipotesi induttiva} \end{aligned}$$

2.8.2 La funzione di Ackermann è calcolabile

Intuizione: per calcolare $\psi(x,y)$ uso $\psi(x',y')$ per un numero finito di coppie (x',y').

Insieme valido: $S \subseteq \mathbb{N}^3$, $(x,y,z) \in S \Rightarrow z = \psi(x,y)$ e se $(x,y,\psi(x,y)) \in S$, allora S contiene anche tutte le triple $(x',y',\psi(x',y'))$ necessarie per calcolare $\psi(x,y)$

Quindi $S \subseteq \mathbb{N}^3$ è valido se

- $(0,y,z) \in S \Rightarrow z = y+1$
- $(x+1,0,z) \in S \Rightarrow (x,1,z) \in S$
- $(x+1,y+1,z) \in S \Rightarrow$ esiste $u \in \mathbb{N}$ tale che $(x+1,y,u)$ e (x,u,z) sono in S.

Si può quindi dimostrare che $\psi(x,y) = z$ se e solo se esiste un insieme di triple $S \subseteq \mathbb{N}^3$ valido e finito tale che (x,y,z) appartiene a S.

Ogni tripla *(x,y,z)* sappiamo che può essere codificata come un numero. Quindi l'insieme S che contiene le varie triple può essere codificato come un insieme di numeri, il quale a sua volta può essere rappresentato come un numero, utilizzando i numeri primi.

Su questo è possibile definire

$$\psi(x,y) = \text{"}\mu(S,z).S \text{ valido e } (x,y,z) \in S\text{"}$$

e quindi è calcolabile. Non andiamo più nel dettaglio, ci basta questa dimostrazione a spanne.

2.8.3 La funzione di Ackermann non è primitiva ricorsiva

Per calcolare la funzione di Ackermann abbiamo utilizzato la minimalizzazione illimitata, ma non è detto che non ci siano altri modi per calcolarla.

Per dimostrare che ψ non è primitiva ricorsiva è necessario dimostrare che non è possibile calcolarla senza la minimalizzazione.

Indicando con $\psi_x(y) = \psi(x,y)$ si ha che

$$\begin{aligned}
\psi_{x+1}(y) &= \psi_x(\underbrace{\psi_{x+1}(y-1)}_{\psi_x(\psi_{x+1}(y-2))}) \\
&\dots \\
&= \psi_x^y(\psi_{x+1}(0)) \\
&= \psi_x^y(\psi(1)) = \psi_x^{y+1}(1)
\end{aligned}$$

$$\begin{aligned}
\psi_0(y) &= y+1 \\
\psi_1(y) &= \psi_0^{y+1}(1) = y+2 \sim y+1 \\
\psi_2(y) &= \psi_1^{y+1}(1) = 2y+3 \sim 2y \\
\psi_3(y) &= \psi_2^{y+1}(1) \sim 2^y \\
\psi_4(y) &= \psi_3^{y+1}(1) \sim 2^{\overbrace{2^{\cdots}}^{y}}
\end{aligned}$$

Il parametro y specifica quindi il numero di iterazioni da fare. es: $\psi_4(3) \approx 2^{2^{2^{16}}}$.

Sia $f \in \mathcal{PR}$ e $k = \#$ *numero di passi di ricorsione primitiva per f.* Allora $P(\vec{x}) \downarrow$ in #passi $< \psi_{k+1}(\max\{\vec{x}\})$.

Si riesce a dimostrare anche che $f(\vec{x}) < \psi_{k+2}(\max\{\vec{x}\})$, noi non lo facciamo, ma questo ci basta per dimostrare che la funzione di Akcermann non è primitiva ricorsiva.

Assumiamo per assurdo che $\psi \in \mathcal{PR}$, allora sia $n = \#$ *numero di annidamenti di ricorsione primitiva nel programma per* ψ.

$$\psi(n+2, n+2) < \psi_{n+2}(\max\{n+2, n+2\}) = \psi_{n+2}(n+2) = \psi(n+2, n+2)$$

Dal momento che questo numero è illimitato, non è possibile calcolarlo a priori con la ricorsione primitiva, è quindi necessario calcolarlo con la minimalizzazione illimitata.

Si ha quindi che

$$\psi \in \mathcal{R} \cap Tot \text{ e } \psi \notin \mathcal{PR}$$

quindi

$$\mathcal{PR} \subsetneq \mathcal{R} \cap Tot$$

Capitolo 3

Programmi generici - Lezione 18

3.1 Enumerazione dei programmi

Enumerazione delle funzioni calcolabili: Un insieme X è numerabile se esiste una funzione $f : \mathbb{N} \to X$ suriettiva tale che $\underbrace{f(0), f(1), f(2), \ldots}_{\mathbb{N}}$.

3.1.1 Enumerazioni di supporto

Per poter enumerare e quindi codificare dei programmi è necessario prima definire alcune funzioni biunivoche di supporto:

1. $\Pi : \mathbb{N}^2 \to \mathbb{N}$
2. $\nu : \mathbb{N}^3 \to \mathbb{N}$
3. $\tau : \bigcup_{k \geq 1} \mathbb{N}^k \to \mathbb{N}$

1. Π

Già osservata precedentemente, ovvero è la codifica delle coppie:

$$\Pi(x, y) = 2^x(2y + 1) - 1$$
$$\Pi^{-1}(n) = (\Pi_1(n), \Pi_2(n))$$

2. ν

ν può essere definita utilizzando Π:

$$\nu(x_1, x_2, x_3) = \Pi(\Pi(x_1, x_2), x_3)$$
$$\nu'(n) = (v_1(n), v_2(n), v_3(n))$$
$$\nu_1(n) = \Pi_1(Pi_1(n))$$

3. τ

τ può essere definita come $(x_1, \ldots, x_k) \rightsquigarrow \prod_{i=1}^{k} P_i^{x_i} - 1$ ma questa funzione non è iniettiva:

$$(2, 1) = 2^2 \cdot 3^1 - 1$$
$$(2, 1, 0) = 2^2 \cdot 3^1 \cdot 5^0 - 1$$

pertanto è necessario utilizzare

$$\tau(x_1, \dots, x_k) \rightsquigarrow \prod_{i=1}^{k-1} P_i^{x_i} \cdot P_k^{x_{k+1}} - 2$$

Per decodificare un numero n è necessario prima calcolare la lunghezza della decodifica:

$$l(n) = \max k.P_k \text{ divide } (n+2) \text{ con } k \leq n$$

Anche se si tratta di una massimizzazione illimitata è possibile dimostrare che è calcolabile sfruttando la minimalizzazione illimitata, e quindi $l(n) \in \mathcal{PR}$.

$$a(n,i) = \begin{cases} (n+2)_i & \text{se } i < l(n) \\ (n+2)_i - 1 & \text{se } i = l(n) \end{cases}$$

l'inversa di τ può essere definita come:

$$\tau^{-1}(n) = \Big(a\big(n,1\big), a\big(n,2\big), \dots, a\big(n,l(n)\big)\Big)$$

3.1.2 Enumerazione dei programmi

Grazie alle funzioni precedentemente definite è possibile definire delle codifiche biunivoche che permettono di codificare con un numero sia una singola istruzione URM, sia un qualsiasi programma URM in $\mathcal{P}$.

$$\beta : \text{Ist URM} \to \mathbb{N} \quad \text{e} \quad \gamma : \mathcal{P} \to \mathbb{N}$$

β : **Ist URM** $\to \mathbb{N}$

$$\begin{aligned} Z(n) &\to 4 \cdot (n-1) \\ S(n) &\to 4 \cdot (n-1) + 1 \\ T(m,n) &\to 4 \cdot \big(\Pi(m-1, n-1)\big) + 2 \\ J(m,n,t) &\to 4 \cdot \big(\tau(m-1, n-1, t-1)\big) + 3 \end{aligned}$$

I vari -1 servono perché i registri URM partono da 1, mentre la codifica deve partire da 0, altrimenti non sarebbe biunivoca.

La moltiplicazione per 4 serve per "*fare spazio*" nella codifica in modo da continuare ad avere una funzione biunivoca. Allo stesso scopo servono anche le somme finali.

per fare l'inversa di β basta calcolare $r = rm(4,n)$ e $q = qt(4,n)$:

$$\beta^{-1} \begin{cases} Z(q+1) & \text{se } r = 0 \\ S(q+1) & \text{se } r = 1 \\ T\big(\Pi_1(q)+1, \Pi_2(q)+1\big) & \text{se } r = 2 \\ J\big(a(q,1)+1, a(q,2)+1, a(q,3)+1\big) & \text{se } r = 3 \end{cases}$$

Verificare J

In questo modo si ha una codifica per le istruzioni che permette di trasformare una sequenza di istruzioni in una sequenza di numeri

$\gamma : \mathcal{P} \to \mathbb{N}$

$$\gamma(P) = \tau\big(\beta(I_1), \beta(I_2), \dots\big)$$

inversa con:

$$\gamma^{-1}(n) = \tau^{-1}(n) = \Big(a\big(n,1\big), a\big(n,2\big), \dots, a\big(n,l(n)\big)\Big)$$

3.2 Verso il programma 33 e la sua funzione

Con quanto a disposizione si può parlare del programma 33, ottenuto decodificando il numero 33.

Prima di continuare serve della notazione aggiuntiva:

Dato $P \in \mathcal{P}$, $\gamma(P)$ indica il codice di P e prende il nome di **Gödel number**.

Dato n, $\gamma^{-1}(n)$ rappresenta il programma codificato nel numero n, tale programma viene indicato con P_n.

Un esempio è dato da:

```
P:
T(1,2)    --> 4Pi(1-1, 2-1) +2= 10
S(2)      --> 4(2-1) = 4
T(2,1)    --> 4Pi(2-1,1-1) +2= 6
```

$$\begin{aligned}\gamma(P) &= \tau(10,4,6)\\ &= 2^1 0 + 3^4 + 5^{6+1} - 2\\ &= 19.439.999.998\end{aligned}$$

```
P':
S(1)    --> 2
```

$$\gamma(P') = \tau(P) = 2$$

I numeri 19.439.999.998 e 2 rappresentano due programmi che calcolano la stessa funzione successore.

Si può fare anche l'operazione inversa:

```
100
gamma^{-1}(100)  100+2
= 2^1 + 3^1 + 17 ^1
P1^1 P2^1 P3^0 P4^0 P5^0 P6^0 P7^1
1 -> S(1)
1 -> S(1)
0 -> Z(1)
0 -> Z(1)
0 -> Z(1)
0 -> Z(1)
1 -> S(1)
```

Il numero 100 rappresenta quindi il programma che calcola la constante 1.

Dal momento che è possibile codificare i programmi in un numero, viene indotta anche un'enumerazione sulle funzioni calcolate da questi programmi:

Fissata $\gamma : \mathcal{P} \to \mathbb{N}$, si ha:

$$\phi_n^{(k)} = f_{P_n}^{(k)}$$

ovvero $\phi_n^{(k)}$ è la funzione di k argomenti calcolata da $P_n = \gamma^{-1}(n)$. Per questa funzione è possibile definire anche

$$W_n^{(k)} = dom(\phi_n^{(k)}) \subseteq \mathbb{N}^k = \{\vec{x} \mid \vec{x} \in \mathbb{N}^k \text{ tale che } \phi_n^{(k)}(\vec{x}) \downarrow\}$$

$$E_n^{(k)} = cod(\phi_n^{(k)}) = \{y \mid \exists \vec{x}\, \phi_n^{(k)}(\vec{x}) = y\}$$

Ad esempio:

$$\begin{aligned} P : S(1) &\to \gamma(P) = 2 \\ \phi_2^{(1)}(x) &= x + 1 \\ W_2^{(1)} &= \mathbb{N} \\ E_2^{(1)} &= \mathbb{N} - \{0\} \end{aligned}$$

Inoltre, si può definire una sorta di funzione di ordine superiore che, una volta fissato un $k \geq 1$, associa un numero $\mathbb{N}$ ad una funzione in $\mathcal{C}^{(k)} : n \to \phi_n^{(k)}$. Questa funzione è suriettiva perché data $f \in \mathcal{C}^{(k)}$ esiste un programma P che calcola $f = f_P^{(k)} = \phi_{\gamma(P)}^{(k)}$. Questa funzioni non è iniettiva perché una stessa funzione può essere calcolata da più programmi.

Quindi

$$| \mathcal{C}^{(k)} | = | \mathbb{N} | \leq | \mathcal{C} | = | \bigcup_{k \geq 1} \mathcal{C}^{(k)} | = | \mathbb{N} |$$

Verificare

ovvero esistono delle funzioni che non sono calcolabili.

3.3 Diagonale di Cantor

Dato un insieme di oggetti numerabile permette di definire un oggetto simile ma che non appartiene alla numerazione.

$$x_0, x_1, x_2$$

x è diverso da x_i nella componente i-esima.

Ad esempio considerando l'insieme della parti dei numeri naturali $\mathcal{P}(\mathbb{N}) = 2^{\mathbb{N}} = \{X \mid X \subseteq \mathbb{N}\}$, questo non è numerabile:

$$| 2^{\mathbb{N}} | > | \mathbb{N} |$$

ovvero non tutti gli infiniti sono uguali.

Questo si dimostra cercando una funzione iniettiva da $\mathbb{N} \to 2^{\mathbb{N}}$, come $x \to \{x\}$, ma noi vogliamo trovare una funzione per la quale non valga l'uguale, perché $2^{\mathbb{N}}$ non è numerabile.

Per assurdo supponiamo che $2^{\mathbb{N}}$ sia numerabile, allora

$$\overbrace{X_0, X_1, X_2, \ldots}^{2^{\mathbb{N}}}$$

```
Si -> il numero appartiene alla partizione X_i
costruita in modo che
N\2^N X0 X1 X2 ....
0 Si    Si No
1 No       No     Si
2 Si        No    No
3 .........
```

Si può quindi considerare la diagonale della tabella ...

```
  X= \{i | i \notin X_i \} \subseteq \mathbb{N} \in 2^\mathbb{N}

  supponendo che esiste i0 in n tale che X = X_i0,
  se i0 \in X_io allora i0 not in X = X_i0, assurdo
  se i0 \notin Xio allora io \in X  = X_i0, assurdo

pertanto 2^\mathbb{N} non è numerabile
```

Altro insieme di non numerabile

```
F-Figa = {f | f : \mathbb{N} \rightarrow \mathbb{N}}

|F-figa| > |N|

\{ f | f : \mathbb{N} \rightarrow {0,1}\} \subseteq F-Figa

|F-figa| >= |\{ f | f : \mathbb{N} \rightarrow {0,1}\} | = |2^\mathbb{N}| > |\mat

|F-figa| > |\mathbb{N}|

consideriamo una qualunque enumerazione degli elementi di F-Figa, vogliamo dimost
figa che non compare nell'enumerazione.

Prendiamo una di queste enumerazioni

N f0  f1  f2 ....
0 f0(0) f1(0)              (funzioni viste come l'insieme dei valori che prendono
1 f0(1)
2

Con questa definizione è possibile costruire una nuova funzione prendendo i valor

f(n) = \uparrow se f_n(n) è \downarrow
           0 se f_n(n) è \uparrow

Per costruzione, questa funzione differisce da ciascuna delle fi riportate in tab

\forall n f_n \neq f, perché f_n(n)\downarrow <==> f(n) \uparrow

allora l'enumerazione non contiene tutte le funzioni in F-figa, ovvero F-
figa non è numerabile.
```

3.3.1 Esistenza di una funzione totale non calcolabile

Esiste una funzione totale non calcolabile:

$$f(n) = \begin{cases} \phi_n(n) + 1 & \text{se } n \in W_n \\ 0 & \text{se } n \notin W_n \end{cases}$$

Per costruzione e per quanto visto prima questa funzione è totale perché è sempre definita e non è calcolabile perché prendendo un qualsiasi enumerazione delle funzioni calcolabili è possibile "*maltrattare*" la diagonale in modo simile a quanto precedente fatto, così facendo si costruisce una funzione che è diversa da tutte le funzioni calcolabili.

Il tutto sta in piedi perché $n \in W_n$ non è decidibile dato che $n \in W_n \Leftrightarrow (\phi_n \downarrow$ e $P_n(n)$ termina).

3.3.2 Esercizio - Diagonalizzazione

Sia $f : \mathbb{N} \to \mathbb{N}$ parziale e sia $k \in \mathbb{N}$. Mostrare che esiste $h : \mathbb{N} \to \mathbb{N}$ non calcolabile, totale e tale che $h(x) = f(x)$ per $x < k$.

Soluzione

```
phi0  phi1 phi2
0 phi0(0) phi1(0)
1       phi0(1)         phi1(1)

k-1   phi0(k-1)  phi1(k-1)
k        phi0(k) phi1(k)
k+1       phi0(k+1) phi1(k+1)

in questo caso si prende la diagonale a partire da k, ovvero

h(x) = \begin{cases}
f(x) se x < k \\
phi_{x-k}(x) +1 se x \geq k e phi_{x-k}(x)\downarrow \\
0 se x \geq k e phi_{x-k}(x)\uparrow \\
\end{cases}

per costruzione h coincide con f fino a k e non è calcolabile perché per ogni phi

h(n+k) = phi_n(n+k)

osservazione: se non ci fosse x-k nella definizione, la funzione h sarebbe ancora

Più formalmente sia g: N -> N, allora esiste e >= k : phi_e = g perché una funzio
```

3.3.3 Esercizio - Diagonalizzazione 2

Data $f : \mathbb{N} \to \mathbb{N}$ si vuole definire $h : \mathbb{N} \to \mathbb{N}$ non calcolabile che coincide con f sui numeri pari.

Soluzione

```
phi0  phi1 phi2
0 phi0(0) phi1(0) FISSO
1       phi0(1)         phi1(1) !=
2       phi0(2)         phi1(2) FISSO

k-1   phi0(k-1)  phi1(k-1)
k        phi0(k) phi1(k)
k+1  phi0(k+1) phi1(k+1)
```

$$h(x) = \begin{cases} f(x), & \text{se } x\text{è pari} \\ \phi_{\frac{x-1}{2}}(x) + 1 & \text{se } x\text{è dispari e} \phi_{\frac{x-1}{2}}(x) \downarrow \\ 0 & \text{se } x\text{è dispari e} \phi_{\frac{x-1}{2}}(x) \uparrow \end{cases}$$

Così facendo h è uguale ad f se x è pari e risulta non essere calcolabile sui numeri dispari perché $\forall n \phi_n(2n+1) \neq h(2n+1)$. È necessario vincolare il $\forall n$ per evitare problemi di codifica.

3.4 Teorema S-M-N

Data una funzione $f : \mathbb{N}^2 \to \mathbb{N}$ calcolabile.

Indiciamo con $f_x : \mathbb{N} \to \mathbb{N}$:

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

Dato che f è calcolabile si ha

$$f = f_p^{(2)} \text{ per un programma } P \ = \phi_e^{(2)} \text{ con } e = \gamma(P)$$

e

$$f_x = \phi^{(1)}_{S(e,x)}$$

ovvero la funzione f_x viene calcolata da un programma che dipende dal programma di indice e, ovvero quello che calcola f e dal valore fissato x.

Ad esempio:

$$\begin{aligned} f(x, y) &= y^x \\ f_x(y) &= y^x \\ f_0(y) &= 1 \\ f_1(y) &= y \\ f_2(y) &= y^2 \\ &\vdots \end{aligned}$$

La funzione $e, x \to S(e, x)$ dati e e x ritorna il programma che calcola f con x fissato.

$S : \mathbb{N}^2 \to \mathbb{N}$ è totale e calcolabile.

$$\phi^{(1)}_{S(e,x)}(y) = \phi^{(2)}_e(x, y)$$

```
e ed x dati

P_e = \gamma^{-1}(e)   |x|y|..... --> phi_e(x,y)

si vuole ottenere l'indice di un programma che calcola phi_e(x,y) per x fissato

P_{e,x} |y|..... --> phi_e(x,y)

per fare questo basta che il programma:
copi y sul secondo registro: |y|y|.....
metta x sul primo registro: |x|y|...
esegue P
```

```
ovvero, P_{e,x} è definito come
T(1,2)
Z(1)
S(1)
...
S(1)
\gamma^{-1}(e)

Pertanto S(e,x) = gamma(P_{e,x})
```

Tutto questo può essere maggiormente formalizzato.

Il nome del teorema deriva dal nome della funzione (S) e dal fatto che si fissano i primi m parametri di una funzione con $m+n$ parametri

3.4.1 Teorema S-M-N (Kleene)

Dato $m, n \geq 1$ esiste $S : \mathbb{N}^{m+1} \to \mathbb{N}$ calcolabile e totale, tale che:

$$\phi_e^{(m+n)}(\vec{x}, \vec{y}) = \phi_{S(e,\vec{x})}^{(n)}(\vec{y})$$

Dimostrazione

Come si calcola $S(e, \vec{x})$?

$P_e = \gamma^{-1}(e)$

```
 1   n
|  y   |.....

P:
T([1..n], [m+1,m+n])
Z(1)
S(1)
...
S(1) --> x1
....
Z(n)
S(n)
...
S(n) --> xn
P_e

S(e,\vec{x}) = \gamma(P)
```

Per dimostrare un po' più formalmente che questa funzione è calcolabile è necessario dimostrare la calcolabilità di alcune funzioni ausiliarie.

$seq : \mathbb{N}^2 \to \mathbb{N}$

$seq(e_1, e_2) = \gamma\left(\frac{P_{e_1}}{P_{e_2}}\right)$

$$ogg : \mathbb{N}^2 \to \mathbb{N}$$

$agg(e, t)$ = codice di $P_e = \gamma^{-1}(e)$ dove ogni `J(m,n,q)` viene sostituita con `J(m,n,q+t)`.

$\widetilde{agg} : \mathbb{N}^2 \to \mathbb{N}$

$$\widetilde{agg}(i,t) = \text{codice dell'istruzione ottenuta aggiornando } \beta^{-1}(i)$$
$$= \begin{cases} i & \text{se } rm(4,i) \neq 3 \\ \nu(\nu_1(q), \nu_2(q), \nu_3(q)+t) \cdot 4 + 3, q = qt(4,i) & \text{altrimenti} \end{cases}$$

Ovvero una volta definita la funzione che aggiorna la codifica dell'istruzione di salto è possibile definire la funzione agg la quale a sua volta permette di definire

$$seq(e_1, e_2) = \gamma(\gamma^{-1}(e_1, agg(\gamma^{-1}(e_2), l(e_1))))$$

dove l calcola la lunghezza del programma. Così facendo si definisce seq come composizione di funzioni calcolabili e totali.

Ci sarebbero poi da definire delle funzioni per le altre istruzioni URM, come:

$$tras(m,n) = \gamma(\text{T[1..n, m+1..m+n]})$$

$$set(i,x) = \gamma(\text{Z(i), S(i) ...}) \text{ setta il valore di } i \text{ a } x$$

Resta poi da definire la funzione $pref_{m,n}$ la quale fornisce il codice aggiunto al programma P_e:

$$pref_{m,n}(\vec{x}) = seq(trasf(m,n), seq(set(1,x_1), seq(\ldots)\ldots)$$

Così facendo

$$S(e,\vec{x}) = seq(pref_{m,n}(\vec{x}), e)$$

3.4.2 Esercizio

Dimostrare che esiste una funzione **calcolabile e totale** $S : \mathbb{N}^2 \to \mathbb{N}$ tale che:

$$\phi_{S(n)}(x) = \lfloor \sqrt[n]{x} \rfloor$$

Soluzione

Se dimostriamo che $f(n,x) = \lfloor \sqrt[n]{x} \rfloor$ è calcolabile possiamo usare SMN per trovare ϕ

$$\begin{aligned} f(n,x) &= \lfloor \sqrt[n]{x} \rfloor \\ &= \max z.\text{"}z^n \leq x\text{"} \\ &= \mu z.\text{"}(z+1)^n > x\text{"} \end{aligned}$$

queste sono minimizzazioni di predicati, quindi la notazione non è propriamente corretta, sarebbe meglio specificare la funzione caratteristica del predicato $P(z,n,x) = (z+1)^n > x$:

$$\mathcal{X}_P(z,n,x) = sg((z+1)^n \dot{-} x)$$

la funzione caratteristica è calcolabile perché deriva dalla composizione di funzioni calcolabile quindi

$$f(n,x) = \mu z.| : \mathcal{X}_P(z,n,x) - 1 |$$

è calcolabile.

Per il teorema SMN esiste una qualche funzione $k : \mathbb{N} \to \mathbb{N}$ tale che

$$\phi_{k(n)}(x) = f(n,x) = \lfloor \sqrt[n]{x} \rfloor$$

Capitolo 4

Esercizi in preparazione al parziale

4.1 Teorema SMN

4.1.1 Esercizio 1

Dimostrare che esiste $S : \mathbb{N} \to \mathbb{N}$ tale che $| W_{S(x)} | = 2x$ e $| E_{S(x)} | = x$

Soluzione

Si cerca una funzione $f(x, y)$ che con x fissato abbia esattamente le caratteristiche richieste, ci pensa il teorema SMN a fare il resto.

$$f(x,y) = \begin{cases} y \dot{-} x, & \text{se } y < 2x \\ \uparrow, & \text{altrimenti} \end{cases}$$
$$= y \dot{-} x + \underbrace{\mu z.y + 1 - 2x}_{\text{fa divergere se } y \geq 2x}$$

Questa funzione è calcolabile e quindi per il teorema SMN esiste $S : \mathbb{N} \to \mathbb{N}$ tale che

$$\phi_{S(x)}(y) = f(x, y)$$

e per costruzione $W_{S(x)} = [0, 2x - 1]$ e $E_{S(x)} = [0, x - 1]$.

4.2 Calcolabilità delle funzioni

4.2.1 Esercizio 1

Dimostrare che

$$f(x) = \begin{cases} \phi_x(x) & \text{se} \phi_x(x) \downarrow \\ 0 & \text{altrimenti} \end{cases}$$

non è calcolabile.

Soluzione

Non si può usare il *trick* della diagonale, perché la funzione deve essere proprio uguale alla diagonale.

$$h(x) = f(x) + 1 = \begin{cases} \phi_x(x) + 1 & \text{se} \phi_x(x) \downarrow \\ 1 & \text{altrimenti} \end{cases}$$

Così facendo $h \neq \phi_x(x) \forall x$ e pertanto non è calcolabile.

Ma h è il successore di f, quindi f non può essere calcolabile, perché se così non fosse $h(x) = f(x) + 1$ sarebbe calcolabile per composizione di funzioni calcolabili.ò

Da notare che questa cosa non vale in se e solo se:

$$h(x) = f(x) + 1 \text{ non calcolabile } \Rightarrow f(x) \text{ non calcolabile}$$

ma **NON È VERO**

$$f(x) \text{ non calcolabile } \Rightarrow f(x) = h(x) + 1 \text{ non calcolabile}$$

4.2.2 Esercizio 2

Sia $\mathbb{P}$ l'insieme dei numeri pari, dimostrare che la funzione caratteristica è primitiva ricorsiva.

$$\mathcal{X}_{\mathbb{P}} = \begin{cases} 1 & \text{se } x \text{ è pari} \\ 0 & \text{altrimenti} \end{cases}$$

Soluzione

$$\mathcal{X}_{\mathbb{P}}(0) = 1 \qquad \mathcal{X}_{\mathbb{P}}(x+1) = \overline{sg}\big(\mathcal{X}_{\mathbb{P}}(x)\big)$$

Dal momento che in un esercizio non sappiamo dell'esistenza del segno negato è necessario andare a mostrare che anche quella funzione è primitiva ricorsiva:

$$\overline{sg}(0) = 1$$
$$\overline{sg}(x+1) = 0$$

4.2.3 Esercizio 3

Dimostrare che la funzione $half(x) = \lfloor \frac{x}{2} \rfloor$ è primitiva ricorsiva.

Soluzione

$$half(0) = 0$$
$$half(x+1) = half(x) + rm_2(x)$$

come prima bisogna dimostrare che rm_2 è ricorsiva primitiva

$$rm_2(x) = 0 \qquad rm_2(x+1) = \overline{sg}(rm_2(x))$$

ma anche $\overleftarrow{sg}$ è in $\mathcal{PR}$

$$\overline{sg}(0) = 1$$
$$\overline{sg}(x+1) = 0$$

4.3 Macchina URM

4.3.1 Esercizio 1

Consideriamo la macchina URM$^-$ che al posto dell'istruzione successore ha quella che effettua il predecessore. Come sono in relazione le due classi?

Ovvero:

$$\mathcal{C}^- \ ? \ \mathcal{C}$$

Soluzione

È chiaro che $\mathcal{C}^- \subseteq \mathcal{C}$ perché la funzione predecessore può essere sostituita da un programma che calcola il predecessore.

Per vedere **se** $\mathcal{C}^- \supseteq \mathcal{C}$ bisogna provare a simulare il successore con un programma per la macchina URM$^-$.

Ma dato un $P \in$ URM$^-$ qualsiasi $P(\vec{x})$ dopo n passi, $\forall n$, tutti i registri $r_i \leq \max x_i$.

Se $n = 0$ è ovvio perché i registri non cambiano.

Se $n \to n+1$, l'istruzione aggiunta può essere:

- precessore ma il valore non aumenta
- zero azzera un registro
- il trasferimento non incrementa i valori

Dal momento che i registri di un programma in URM$^-$ non aumentano mai, non è possibile implementare la funzione successore, quindi **non vale** la relazione $\mathcal{C}^- \supseteq \mathcal{C}$.

4.3.2 Esercizio 2

URM^S che non ha `S(n)` e `J(m,n,t)` ma ha `JS(m,n,t)` che prima fa il test e poi incrementa il registro m.

Soluzione

$\mathcal{C}^S \subseteq \mathcal{C}$ perché un programma URM riesce ad emulare l'istruzione `JS`.

$\mathcal{C}^S \supseteq \mathcal{C}$ non vale, perché un programma URM^S non riesce a calcolare la funzione successore.

4.4 Diagonalizzazione

4.4.1 Esercizio 1

$F_0 = \{f \mid \text{cod}(f) \subseteq \{0\} \text{ e parziali}\}$ è numerabile?

Soluzione

L'unica differenza tra queste funzioni è data dai domini che sono tanti quanti i numeri naturali.

Suppongo per assurdo che F_0 sia enumerabile, allora:

```
f0 f1 f2
0       0    0    0
1       0    0
2
```

considerando la diagonale, posso definire

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

f dovrebbe appartenere a F_0 ma per costruzione non compare nella enumerazione, quindi F_0 non è enumerabile.

4.4.2 Esercizio 2

$f : \mathbb{N} \to \mathbb{N}$ totale e decrescente se $\forall x, y \; x \leq y \Rightarrow f(y) \leq f(x)$.

$F_d = \{f : \mathbb{N} \to \mathbb{N} \quad f \text{ totale e descrescente e } \text{cod}(f) \subseteq \{0, 1\}\}$ è numerabile?

Soluzione

Le funzioni dell'insieme possono essere o costanti uguali a 0 o 1, oppure uguali a 1 fino ad un certo valore per poi diventare costanti a 0.

$$\forall f \in F_d \, n(f) = \sup(\{x \mid f(x) = 1\})$$

$n(f)$ è il punto in cui cambia il valore della funzione e caratterizza f, ovvero

$$\forall f_1, f_2 \in F_d \, n(f_1) = n(f_2) \Rightarrow f_1 = f_2$$

quindi $n : F_d \to \mathbb{N} \cup \{\infty\}$ è una funzione iniettiva e suriettiva, pertanto $\mid F_d \mid = \mid \mathbb{N} \cup \{\infty\} \mid$, ovvero F_d è numerabile.

4.4.3 Esercizio 3

Trovare $f : \mathbb{N} \to \mathbb{N}$ totale e non calcolabile tale che $f(x) = x$ per infiniti x.

Soluzione

$$f(x) = \begin{cases} x, & \text{se } x \text{ è pari} \\ \phi_{\frac{x-1}{2}}(x) + 1, & \text{se } \phi_{\frac{x-1}{2}}(x) \downarrow \\ 0, & \text{se } \phi_{\frac{x-1}{2}}(x) \uparrow \end{cases}$$

4.4.4 Esercizio 4

Trovare una funzione f crescente, totale e non calcolabile.

Soluzione

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

h è non calcolabile perché è definita sulla diagonale distorta.

$$f(x) = \sum_{y \leq x} h(y)$$

Così definita f è:

- totale
- crescente: $f(x+1) = f(x) + h(x+1) \geq f(x)$
- $\forall x$ se $\phi_x(x) \downarrow$ $f(x) = \sum_{y \leq x} h(y) \geq h(x) = \phi_x(x) + 1 \neq \phi_x(x)$ e se $\phi_x(x) \uparrow$, $h(x) = 0 \neq \phi_x(x)$, ovvero f non è calcolabile.

4.4.5 Esercizio 5

Può esistere $f : \mathbb{N} \to \mathbb{N}$ non calcolabile tale che $\forall g : \mathbb{N} \to \mathbb{N}$ non calcolabile $f + g$ è calcolabile?

$$(f + g)(x) = f(x) + g(x)$$

Soluzione

Non può esistere, perché per assurdo, sia f una funzione tale che $(f + f)$ è calcolabile:

$$\begin{aligned} h(x) &= f(x) + f(x) \\ &= 2f(x) \end{aligned}$$

$h(x)$ è quindi calcolabile, però utilizzando questa funzione si può definire $f(x) = qt(2, h(x))$, dimostrando la calcolabilità di f che per ipotesi non è calcolabile.

4.5 Correzione compitino

4.5.1 Esercizio 1

Definire la classe delle funzioni ricorsive e dimostrare che $pow2(y) \in \mathcal{PR}$

Soluzione

La definizione è negli appunti.

Per dimostrare che la funzione è primitiva ricorsiva la si definisce per ricorsione primitiva:

$$\begin{cases} pow2(0) & = 1 \\ pow2(y+1) & = double(pow2(y)) \end{cases}$$

Serve definire anche *double* come primitiva ricorsiva:

$$\begin{cases} double(0) & = 0 \\ double(y+1) & = S(S(double(y))) \end{cases}$$

*pow*2 è quindi primitiva ricorsiva.

4.5.2 Esercizio 2

Analizzare la classe delle funzioni calcolabili $\mathcal{C}'$ associate alla macchina URM' che al posto dell'istruzione di salto e dell'istruzione successore ha l'istruzione `JI(m,n,t)` che se trova i due registri uguali, prima incrementa il registro `m` e poi effettua il salto all'istruzione `t`.

Soluzione

Le istruzioni `J` e `S` della macchina URM possono essere replicate con

```
// J(m,n,t)
T(m,k)
JI(k,n,t)
// S(n)
Il: JI(n,n,l+1)
```

Serve poi la prova induttiva sul numero di istruzioni che è sempre quella, l'importante è specificare che nei passi intermedi della dimostrazione si ottiene un programma ibrido.

C'è poi da fare la dimostrazione dell'altro verso, che avviene in modo analogo.

4.5.3 Esercizio 3

Può esistere una funzione f totale e non calcolabile tale che $cod(f) = \mathbb{P}$.

Soluzione

Perché sia totale basta che sia sempre definita, per ottenere la non calcolabilità la devo rendere diversa da tutte le funzioni calcolabili.

$$f(x) = \begin{cases} x, & \text{se } x \text{ è pari} \\ 2(\phi_{\frac{x-1}{2}}(x) + 1), & \text{se } x \text{ è disparie e} \downarrow \\ 0, & \text{altrimenti} \end{cases}$$

Ma anche una definizione più semplice va bene:

$$f(x) = \begin{cases} 2(\phi_x(x) + 1), & \text{se } \phi_x totale \\ 0, \text{ altrimenti} \end{cases}$$

Capitolo 5

Funzione universale

Vogliamo un programma universale che prende in input il numero del programma e l'input da utilizzare e fornisce in output il risultato dell'esecuzione del programma sul dato input.

$$\Psi_U(x, y) = \phi_x(y)$$

La funzione universale ha in generale $k + 1$ argomenti:

$$\Psi_U^{(k)} : \mathbb{N}^{k+1} \to \mathbb{N}$$
$$\Psi_U(e, \vec{x}) = \phi_e^k(\vec{x})$$

5.1 Calcolabilità della funzione universale

Per calcolare $\Psi_U(e, \vec{x}) = \phi_e^k(\vec{x})$ è necessario come prima cosa recuperare il programma rappresentato da e utilizzando $\gamma^{-1}(e) = P_e$.

Dopodiché è necessario eseguire $P_e(\vec{x})$, la quale può produrre un output $\phi_e^k(\vec{x}) = \Psi_U(e, \vec{x})$ oppure può non terminare e in questo caso $\Psi_U(e, \vec{x}) \uparrow$.

5.1.1 Dimostrazione pseudo-formale

`|r1|r2|r3|...|0|...`

$C = \prod_{i \geq 1} = p_i^{r_i}$ con questa codifica della memoria è possibile accedere al contenuto del registro i-esimo con $r_i = (C)_i$.

Qualcosa sulla codifica dello stato del programma: s$\sigma = \Pi(C, J)$.

$C_k(e, \vec{x}, t)$ = contenuto dei registri dopo t passi del programma P_e sull'input $\vec{x}$.

$J_k() = \begin{cases} \text{istruzione da eseguire dopo di passi di } P_e(\vec{x}), \text{ se non termina.} \\ 0 \text{ altrimenti} \end{cases}$

C'è da dimostrare che $C_k, J_k \in \mathcal{PR} \subseteq \mathcal{R} = \mathcal{C}$, una volta fatto ciò è possibile definire:

$$\Psi_U(e, \vec{x}) = \phi_e^k(\vec{x}) = \Big(C_k(e, \vec{x}, \mu t. J_k(e, \vec{x}, t))\Big)_1 \in \mathcal{R} = \mathcal{C}$$

Sulla base di $C_p(\vec{x}, t)$ e $J_p(\vec{x}, t)$ si possono definire per ricorsione primitiva:

$$C_p()$$

Servono prima un po' di funzioni ausiliarie.

Argomenti di un istruzione URM $i = \beta(\text{Istr})$

$$\begin{aligned} Z_{arg}(i) &= qt(4,i)+1 \\ S_{arg}(i) &= qt(4,i)+1 \qquad\qquad i=(n-1)\times 4 \\ T_{arg_1} &= \Pi_1(qt(4,i))+1 \qquad\quad i=\dots \\ T_{arg_2} &= \Pi_2(qt(4,i))+1 \end{aligned}$$

Effetto dell'esecuzione di un istruzione:

$$zero(C,n) = qt(p_n^{(C)_n}, C)$$

$$succ(C,n) = p_n \cdot C$$

$$trasf(C,m,n) = zero(C,n)\cdot p_n^{(C)_m}$$

Effetto dell'esecuzione di un istruzione $i = \beta(\text{Itrs})$:

$$change : \mathbb{N}^2 \to \mathbb{N}$$

$$change(C,i) = \begin{cases} zero(C, Z_{arg}(i)) se\, rm(4,i) = 0 \\ succ(C, S_{arg}(i)) se\, rm(4,i) = 1 \\ trasf(C, T_{arg_1}(i), T_{arg_2}(i)) serm(4,i) = 2 \\ Caltrimenti(ilsaltononcambialaconfigurazionedimemoria) \end{cases}$$

Prossima configurazione di memoria

$nextconf(e,C,t) =$ Configurazione dei registri al prossimo passo partendo da C e eseguendo la t-esima istruzion
$= change(C, a(e,t))$

dove a è la funzione che data la codifica di un programma ritorna il codice della t-esima istruzione.

$is(C,i,t) =$ numero dell'istruzione successiva se al passo corrente eseguo $i = \beta(\text{Istr})$ che si trova in posizione t del programma.

$$is(C,i,t) = \begin{cases} t+1, & \text{se } rm(4.i) \neq 3 \text{ oppure } (C)_{J_{arg_1}(i)} \neq (C)_{J_{arg_2}(i)} \\ J_{arg_3}(i), & \text{altrimenti} \end{cases}$$

$nextistr(e,C,t) =$ prossima istruzione a partire da C ed eseguendo t

$$nextistr(e,C,t) = \begin{cases} is(C, a(e,t), t), & \text{se } is(C,a(e,t),t) \neq l(e)+1 \\ 0, & \text{altrimenti} \end{cases}$$

Siamo quasi alla fine

$$\begin{aligned} C_k(e,\vec{x},0) &= \prod_{i=1}^{k} p_i^{r_i} \\ J_k(e,\vec{x},0) &= 1 \\ C_k(e,\vec{x},t+1) &= nextconf(e, C_k(e,\vec{x},t), J_k(e,\vec{x},t)) \\ J_k(e,\vec{x},t+1) &= nextistr(e, C_k(e,\vec{x},t), J_k(e,\vec{x},t)) \end{aligned}$$

Dato che tutte le funzioni utilizzate, anche C_k e J_k sono primitive ricorsive e quindi calcolabili.

$$\Psi_U^{(k)} = \Big(C_k(e,\vec{x},\mu t.J_k(e,\vec{x},t))\Big)_1 \in \mathcal{R}$$

5.2 Corollari-rio

Sono decidibili i seguenti predicati

1. $H_k(e, \vec{x}, t) = P_e(\vec{x}) \downarrow$ in t o meno passi.
2. $S_k(e, \vec{x}, y, t) = P_e(\vec{x}) \downarrow y$ in t o meno passi.

Entrambi sono calcolabili perché basta fare andare il programma per t passi e osservare l'output.

$$\mathcal{X}_{H_k}(e, \vec{x}, t) = \overline{sg}(J_k e, \vec{x}, t)$$

$$\mathcal{X}_{S_k}(e, \vec{x}, y, t) = \overline{sg}((J_k(e, \vec{x}, t)) + \mid y - (C_k(e, \vec{x}, t))_1 \mid)$$

5.3 Funzione inversa con i corollari

Data una funzione $f : \mathbb{N} \to \mathbb{N}$iniettiva, totale e calcolabile. Allora

$$f^{-1}(y) = \begin{cases} x \text{ tale che } f(x) = y \\ \uparrow \text{ altrimenti} \end{cases}$$

è calcolabile con

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

Però se f non è totale, non è più garantita la calcolabilità.

L'idea è quindi quella di eseguire un certo numero di passi via via crescente su ogni programma. (In teoria questa cosa è già stata fatta. Però non si poteva spiegare in modo formale).

Essendo f calcolabile questa sarà calcolata da un certo programma e.

$$f^{-1}(y) = \mu(x, t).S(e, x, y, t)$$

Peccato che non esiste un operatore che minimizza le coppie. Neanche la minimalizzazione innestata funziona, perché scorrerei la tabella prima solo sulle colonne e poi solo sulle righe.

Serve quindi un barbatrucco, in modo da riuscire a codificare una coppia con un numero intero.

$$\mu w.S(e, (w)_1, y, (w)_2)$$

Ovvero del numero w ci interessano solo gli esponenti dei primi due divisori primi del numero. Piccola nota: usiamo un predicato nella minimalizzazione quando sarebbe necessario utilizzare una funzione. Sarebbe quindi più corretto utilizzare il valore assoluto della funzione caratteristica del predicato, meno uno.

$$f^{-1}(y) = \Big(\mu w.S(e, (w)_1, y, (w)_2)\Big)$$

5.4 Cose non calcolabili

Ci sono un sacco di cose che non sono calcolabili e un sacco di cose non decidibili.

5.4.1 Totalità di un programma

Predicato $Tot(x) = \phi_x$ è totale non è decidibile.

Dimostrazione

Supponiamo per assurdo che $Tot(x)$ sia decidibile.

Definiamo

$$f = \begin{cases} \phi_x(x) + 1 \text{ se } x \text{ è totale} \\ 1 \text{ altrimenti} \end{cases}$$

Questa funzione è certamente totale e $f \neq \phi_x \forall x$ tale che ϕ_x è totale perché se ϕ_x è totale $f(x) = \phi_x(x) + 1 \neq \phi_x(x)$ e quindi non è calcolabile.

Sfruttando però la totalità di Tot è possibile scrivere la funzione $\phi_($x$)$ come $\Phi_U(x, x)$ e quindi f risulterebbe definita per casi, utilizzando solamente funzioni calcolabili e quindi anch'essa sarebbe calcolabile. Non è però possibile utilizzare la classica aritmetizzazione dell'*if*, perché quando ϕ_x non è totale si ha un risultato indefinito.

$$f(x) = \left(\mu w.\Big(S(x, x, (w)_1, (w)_2) \wedge Tot(x) \vee (w)_1 = 0 \wedge \neg Tot(x)\Big)\right)_1 + 1$$

Con questa definizione il programma ϕ_x vengono eseguiti sempre un po' di passi alla volta, così quando $Tot(x)$ non è totale viene ritornato il valore corretto.

f risulta quindi calcolabile e questo è assurdo per costruzione di *f*.

5.4.2 Esercizio

Q(x) decidibile, f1,f2 N -> N calcolabili. f(x) = f1(x) se Q(x), f2(x) altrimenti è calcolabile?

TODO

5.4.3 Esercizio - Terminazione del programma sul suo indice

$H(x) = \phi_x(x)\downarrow$ non è decidibile.

5.5 Operazioni effettive su programmi

Dati due programmi P_x, P_y questi calcoleranno le due funzioni ϕ_x, ϕ_y.

$$(\phi_x + \phi_y)(z) = \phi_x(z) + \phi_y(z) = \phi_e(z) = \phi_{k(x,y)}(z)$$

Con K funzione totale e calcolabile.

Ovvero esiste $K : \mathbb{N}^2 \to \mathbb{N}$ tale che $\phi_{k(x,y)}(z) = \phi_x(z) + \phi_y(z)$.

Questo si dimostra prima utilizzando il teorema SMN:

$$g(x, y, z) = \phi_x(z) + \phi_y(z) = \Phi_U(x, z) + \Phi_U(y, z)$$

è calcolabile e quindi per il teorema SMN esiste $K : \mathbb{N}^2 \to \mathbb{N}$ totale e calcolabile tale che

$$g(x, y, z) = \phi_{K(x,y)}(z)$$

5.5.1 Funzione inversa

Un ragionamento simile può essere fatto con la funzione inversa.

Esiste $K : \mathbb{N} \to \mathbb{N}$ totale e calcolabile, tale che

$$\phi_{K(x)}(y) = (\phi_x)^{-1}(y)$$

Definisco quindi

$$g(x,y) = (\phi_x)^{-1}(y) = \Big(\mu w.S\big(x,(w)_1,y,(w)_2\big)\Big)_1$$

g è calcolabile e quindi per il teorema SMN si ha che esiste $K : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che

$$g(x,y,z) = \phi_{K(x,y)}(z)$$

5.5.2 Programmi con lo stesso domino

Esiste $K : \mathbb{N}^2 \to \mathbb{N}$ totale e calcolabile, tale che

$$W_{K(x,y)} = W_x \cup W_y$$

ovvero si vuole che il programma composto abbia come dominio l'unione dei due domini.
Serve quindi una funzione per l'esecuzione dei due programmi passo passo:

$$g(x,y,z) = \nVdash(\mu w.(H(x,z,w) \vee H(y,z,w)))$$

questa funzione è definita se z appartiene ad almeno uno dei due domini ed è indefinita se non compare in nessuno dei due domini.

$$g(x,y,z) = \begin{cases} 1 \text{ se } z \in W_x \cup W_y \\ \uparrow \text{ altrimenti} \end{cases}$$

Essendo questa funzione calcolabile, per il teorema SMN esiste una funzione $K : \mathbb{N}^2 \to \mathbb{N}$ totale e calcolabile tale che

$$\phi_{K(x,y)}(z) = g(x,y,z)$$

Così facendo si ottiene proprio la funzione desiderata perché

$$z \in W_{K(x,y)} \Leftrightarrow \phi_{K(x,y)}(z) \downarrow \Leftrightarrow z \in W_x \cup W_y$$

5.5.3 Programmi con gli stessi codomini

Esiste $S : \mathbb{N}^2 \to \mathbb{N}$ totale e calcolabile tale che

$$E_{S(x,y)} = E_x \cup E_y$$

L'idea è quella di eseguire P_x se z è pari altrimenti se è dispari viene eseguito P_y. I due programmi devono però essere eseguiti su $z/2$ in modo che possano essere eseguiti su tutti i numeri.

$$g(x,y,z) = \Big(\mu w.\Big(\big(Pari(z) \wedge S(x,z/2,(w)_1,(w)_2)\big) \vee \big(\neg Pari(z) \wedge S(y,z/2,(w)_1,(w)_2)\big)\Big)\Big)_1$$

Si ha quindi che la funzione

$$g(x,y,z) = \begin{cases} \phi_x(z/2) \text{se } z \text{ è pari} \\ \phi_y(z/2) \text{se } z \text{ è dispari} \end{cases}$$

è calcolabile per come è stata precedentemente definita. (Sarebbe da ricopiare la definizione, utilizzando qt al posto della divisione classica).
Essendo calcolabile segue che per il teorema SMN esiste $S : \mathbb{N}^2 \to \mathbb{N}$ calcolabile e totale, tale che

$$\phi_{S(x,y)}(z) = g(x,y,z)$$

ed è la funzione cercata perché:

$$v \in E_{S(x,y)} \Rightarrow \exists z | \phi_{S(x,y)}(z) = v \Rightarrow \begin{cases} g(x,y,z) = v = \phi_x(z/2) \text{ se } z \text{ è pari} \Rightarrow v \in E_x \\ g(x,y,z) = v = \phi_y(z/2) \text{ se } z \text{ è dispari} \Rightarrow v \in E_y \end{cases} \Rightarrow v \in E_x \cup E_y$$

e vale anche l'altra inclusione, perché, sia $v \in E_x$, esiste z tale che $\phi_x(z) = v \Rightarrow \phi_{S(x,y)}(2z) = g(x,y,2z) = \phi_x(z)$ e siccome per un qualche argomento z si ha che $v \in E_{S(x,y)}$.

5.6 Esercizi

1. Esiste $K : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $E_{K(x)} = W_x$?
2. Data $f : \mathbb{N} \to \mathbb{N}$ calcolabile, esiste $K : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $\forall x W_{K(x)} = f^{-1}(W_x)$

Capitolo 6

Insiemi Ricorsivi e Ricorsivamente Enumerabili

Ogni proprietà interessante del comportamento dei programmi non è calcolabile (correttezza, assenza di bug, terminazione).

Dal punto di vista della computabilità tutti i problemi decidibili vengono considerati facili, anche se richiedono un carico computazionale estremamente elevato.

Ci sono poi i problemi semi-decidibili, per i quali si riesce a rispondere solo in caso positivo e i problemi indecidibili per i quali non è sempre possibile fornire una risposta.

6.1 Insiemi

Dato un sottoinsieme $X \subseteq \mathbb{N}$, $x \in X$? Quando questa proprietà è decidibile si dice che l'insieme è **ricorsivo**, mentre se è semi-decidibile, l'insieme è **ricorsivamente enumerabile**.

La definizione utilizza i numeri, ma per noi questi possono essere considerati come dei programmi:

$$X = \{x | P_x \ldots\}$$

6.1.1 Insiemi ricorsivi

$A \subseteq \mathbb{N}$ si dice ricorsivo se la sua funzione caratteristica

$$\mathcal{X}_A : \mathbb{N} \to \mathbb{N}$$

che vale 1 se $x \in A$ e 0 altrimenti è calcolabile, ovvero se il predicato $x \in A$ è decidibile.

$\mathbb{N}$ è ricorsivo, perché la sua funzione caratteristica è la costante 1.

$P_r = \{x | x \text{ è primo}\}$ è anche esso ricorsivo.

Come risultato più generale si ha che tutti gli insiemi $A \subseteq \mathbb{N}$ e finiti sono ricorsivi, perché la loro funzione caratteristica può essere sempre definita come una serie di *if*.

L'insieme $K = \{x | \phi_x(x) \downarrow\}$ non è ricorsivo, perché se per assurdo lo fosse, sarebbe possibile definire una funzione calcolabile diversa da tutte quelle calcolabili:

$$f(x) = \begin{cases} \phi_x(x) + 1 & x \in K \\ 1 & x \notin K \end{cases}$$

la funzione così definita è totale e calcolabile, perché può essere definita come

$$\begin{aligned} f(x) &= (\mu w.(xinK \wedge (w)_1 = \phi_x(x) \wedge (w)_2 = \text{ numero di passi per termianre}) \vee (x \notin K \wedge (w)_1 = 0))_1 + 1 \\ &= (\mu w.(x \in K \wedge S(x, x, (w)_1, (w)_2)) \vee (x \notin K \wedge (w)_1 = 0))_1 + 1 \end{aligned}$$

e per combinazione di funzione calcolabili e minimalizzazione è calcolabile. f risulta quindi calcolabile e questo è un problema perché

$$\forall x \phi_x(x) \begin{cases} x \in K & f(x) = \phi_x(x) + 1 \neq \phi_x(x) \\ x \notin K & f(x) = 1 \neq \phi_x(x) \uparrow \end{cases}$$

quindi K non può essere ricorsivo.
Anche $T = \{x | \phi_x \text{ è totale}\}$ non è ricorsivo.

Chiusura degli insiemi ricorsivi

Se A e B sono due insiemi ricorsivi, anche

- $\bar{A} = \mathbb{N} \; A$
- $A \cup B$
- $A \cap B$

sono ricorsivi.

6.2 Riduzione

Si hanno due problemi, $\mathcal{A}$ e $\mathcal{B}$ e si vuole poter dire se $\mathcal{A}$ è più facile o più difficile di $\mathcal{B}$.

Questo prende il nome di **m-riducibilità**[1] e con questo si intende che dati $A, B \subseteq \mathbb{N}$, il problema $x \in A$ si riduce al problema $x \in B$ (notazione $A \leq_m B$) se esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $\forall x \; x \in A \Leftrightarrow f(x) \in B$.

Ovvero esiste un modo che permette di trasformare un'istanza del problema $\mathcal{A}$ in un'istanza del problema $\mathcal{B}$ tale che se l'istanza x soddisfa $\mathcal{B}$, questa soddisfa anche $\mathcal{A}$.

In questo caso il problema $\mathcal{A}$ è più facile da risolvere perché sapendo risolvere il problema $\mathcal{B}$ si riesce a risolvere anche $\mathcal{A}$.

Più formalmente, se $A, B \subseteq \mathbb{N} \; A \leq_m B$ allora

1. se B è ricorsivo anche A è ricorsivo
2. se A non è ricorsivo, anche B non è ricorsivo

Il punto 1 si dimostra a partire dalla funzione caratteristica di B: se B è ricorsivo,

$$\mathcal{X}_B(x) = \begin{cases} 1 & x \in B \\ 0 & x \notin B \end{cases}$$

la funzione caratteristica di A può essere definita come

$$\mathcal{X}_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} = \mathcal{X}_B(f(x))$$

Il punto 2 si dimostra in modo simile.

L'insieme $T = \{x | \phi_x(x) \text{totali}\}$ non è ricorsivo e può essere dimostrato utilizzando la riduzione $K \leq_m T$[2].

È quindi necessario trovare una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $x \in K$ se e solo se $f(x) \in T$. Vogliamo quindi una funzione che fornisca sempre un output se $x \in K (P_x(x) \downarrow)$ e che non fornisca mai un output se $x \notin K$.

[1] Ometteremo il prefisso m-.
[2] la m si può omettere

Possiamo quindi definire una funzione $g(x, y)$ con x il programma da trasformare e y l'input di tale programma (che non verrà mai usato).

$$g(x,y) = \begin{cases} \phi_x(x) & x \in K \\ \uparrow & \text{altrimenti} \end{cases} = \phi_x(x) = \Phi_U(x,x)$$

essendo g calcolabile, per il teorema SMN esiste una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che

$$g(x,y) = \phi_{f(x)}(y)$$

f è quindi una funzione di riduzione di K a T perché se $x \in K$, $f(x) \in T$ si ha $\phi_{f(x)}(y) = g(x,y) = \phi_x(x) \downarrow \forall y$ totale ($f \in T$), viceversa se $x \notin K$, $f(x) \notin T$ perché $\phi_{f(x)}(y) = g(x,y) = \phi_x(x) \uparrow \forall y$ e non è totale, quindi $f \notin T$.

Quindi $K \leq_m T$ tramite f, pertanto essendo K non ricorsivo neppure T lo sarà.

6.2.1 Problema dell'input

$$A_n = \{x | \phi_x(n) \downarrow\}$$

Per un n fissato, questo insieme non è ricorsivo e lo si dimostra per riduzione con $K \leq_m A_n$.

Serve quindi una funzione S calcolabile e totale, tale che se $x \in K$, $f(x) \in A_n$, ovvero si vuole trasformare un programma P_x in un altro programma che prende in input un valore y e che se $x \in K$ allora $P_{f(x)}(n) \downarrow$ e che se $x \notin K$, $P_{f(x)}(n) \uparrow$.

In modo simile a prima è possibile definire

$$g(x,y) = \phi_x(x) = \Phi_U(x,x)$$

per il teorema SMN esiste quindi $S : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che

$$g(x,y) = \phi_{S(x)}(y)$$

S è la funzione di riduzione di K a A_n perché se $x \in K$, $\phi_{S(x)}(y) = g(x,y) = \phi_x(x) \downarrow$ e in particolare $\phi_{S(x)}(n) = \phi_x(n)$ che è definita e quindi $S(x) \in A_n$.

Se invece $x \notin K$, $\phi_{S(x)}(y) = \phi_x(x) \uparrow \forall y$ e quindi, per definizione di A_n si ha che $S(x) \notin A_n$

6.2.2 Problema dell'output

$$B_n = \{x | n \in E_x\}$$

Per un n fissato, questo insieme non è ricorsivo e lo si dimostra per riduzione con $K \leq_m B_n$.

Serve quindi una funzione f tale che $x \in K \Leftrightarrow f(x) \in B_n$.

La dimostrazione è uguale a quella precedente con la differenza che il programma $P_{f(x)}(x)$ deve sempre ritornare n:

$$g(x,y) = \begin{cases} n & x \in K \\ \uparrow & \text{altrimenti} \end{cases} = n \cdot 1(\phi_x(x))$$

Essendo calcolabile, per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che

$$g(x,y) = \phi_{f(x)}(y)$$

f è la funzione di riduzione di $K \leq_m B_n$ perché se $x \in K$, $\phi_{f(x)}(y) = g(x,y) = n$ che certamente ha n nel suo codominio ($n \in E_{f(x)} = \{n\}$) e quindi $f(x) \in B_n$. Se invece $x \notin K$, $\phi_{f(x)}(y) \uparrow = g(x.y) \forall y$, ovvero $E_{f(x)} = \emptyset$ e quindi $f(x) \notin B_n$.

6.3 Teorema di Rice

Ogni proprietà del comportamento di un programma non è decidibile (es: il programma termina sempre, termina in un certo input, ecc.). Queste proprietà sono quelle che riguardano che cosa calcola il programma, non come è definito.

Un **insieme saturato** (versione formale delle proprietà dei programmi) $A \subseteq \mathbb{N}$ tale che se $x \in A$ e $y \in \mathbb{N}$ e tale che $\phi_x = \phi_y$, allora anche $y \in A$.

A è saturato se e solo se esiste $\mathcal{A} \subseteq \mathcal{C}$ tale che $A = \{x | \phi_x \in A\}$, ovvero il fatto che un programma appartenga ad un insieme saturo non dipende dalla struttura del programma ma dalla funzione che esso calcola.

Ad esempio $T = \{x | \phi_x \text{ è totale}\}$ è un insieme saturo, perché la proprietà che un programma termina è come dire che la funzione calcolata dal programma termina sempre.

Anche $\{x | \phi_x = \lambda y.\sqrt{y}\}$ è saturo, perché contiene tutte le funzioni che calcolano la radice quadrata.

$K = \{x | x \in W_x\}$ non è saturo perché intuitivamente non si riesce definire in modo preciso un insieme di funzioni. L'idea è quella di mostrare che esiste una funzione calcolabile $e \in \mathbb{N}$ tale che

$$\phi_e(x) = \begin{cases} e & x = e \\ \uparrow & \text{altrimenti} \end{cases}$$

per come è definita questa funzione $e \in W_e \Rightarrow e \in K$ ed esiste $e' \in \mathbb{N}$ tale che $\phi_e = \phi_{e'}$ e $e \neq e'$.

Si ha che $\phi_{e'}(e') = \phi_e(e') = \uparrow$ e quindi $e' \notin K$, ovvero l'insieme K non contiene tutti gli indici che calcolano la stessa funzione.

Tutto questo si basa sul fatto che e esiste e arriveremo più avanti a dimostrarlo.

6.3.1 Enunciato del teorema di Rice

Sia A una qualunque proprietà del comportamento dei programmi, ovvero $A \subseteq \mathbb{N}$ e saturo.

Se la proprietà non è banale, l'insieme non è ricorsivo.

Con proprietà non banale si intente $A \neq \mathbb{N}$ (sempre vera) e $A \neq \emptyset$ (sempre falsa).

Dimostrazione

L'obiettivo è quello di dimostrare che $K \leq_m A$.

Per ipotesi sappiamo che A non è vuoto e non è tutto $\mathbb{N}$, quindi è definito anche l'insieme $\bar{A}$. Vogliamo trovare una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che $x \in K \Leftrightarrow f(x) \in A$.

Consideriamo un qualunque indice $e_0 \in \mathbb{N}$ $\phi_{e_0} = \emptyset$ ovvero che non termina mai.

Ci sono due possibili casi:

1. $e_0 \in \bar{A}$: sia $e_1 \in A$ un programma qualsiasi (che sappiamo esistere perché $\bar{A} \neq \emptyset$). Possiamo definire
$$g(x,y) = \begin{cases} \phi_{e_1}(y) & x \in K \\ \phi_{e_0}(y) & x \notin K \end{cases} = \begin{cases} \phi_{e_1}(y) & x \in K \\ \uparrow & x \notin K \end{cases} = \phi_{e_1}(y) \cdot 1(\phi_x(x))$$
che è calcolabile perché può essere scritta utilizzando la funzione calcolabile. Perciò è possibile utilizzare il teorema SMN per avere la garanzia dell'esistenza di una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $g(x,y) = \phi_{f(x)}(y) \forall x, y$, che può essere utilizzata come funzione di riduzione. Questo perché se $x \in K \Rightarrow \phi_{f(x)}(y) = \phi_{e_1}(y) \forall y$ e dato che A è saturato e che $e_1 \in A$, allora anche $f(x) \in A$ perché essendo saturo A contiene tutti gli indici che calcolano quella funzione. Se invece $x \notin K \Rightarrow \phi_{f(x)}(y) = \phi_{e_0} \forall y$ e allo stesso modo dato $e_0 \notin A$ anche $f(x) \notin A$ perché se questo non fosse vero anche e_0 dovrebbe appartenere ad A. Concludendo, dato che K non è ricorsivo e che $K \leq_m A$, anche A non è ricorsivo.

2. $e_0 \in A$: sia $B = \bar{A}$, $B \neq \emptyset \neq \mathbb{N}$ e saturo perché è l'insieme complementare di un insieme saturo ed $e_0 \notin B$. Per quanto dimostrato al punto 1, si ha che B non è ricorsivo e quindi anche $\bar{A}$ non è ricorsivo, allora anche A non è ricorsivo perché è l'insieme complementare di un insieme ricorsivo.

$A_n = \{x | n \in W_x\} = \{x | \phi_x(n) \downarrow\}$ è saturo perché può essere visto come

$$A_n = \{x | \phi_x \in \mathcal{A}_n\}$$
$$\mathcal{A}_n = \{f \in \mathcal{C} | n \in dom(f)\}$$

Si ha poi che

1. $A_n \neq \emptyset$: e_1 tale che $\phi_{e_1} = id$, $e_1 \in A_n$
2. $A_n \neq \mathbb{N}$: e_0 tale che $\phi_{e_0} = \emptyset$, $e_0 \notin A$

Quindi per il teorema di Rice A_n non è ricorsivo.
$B_n = \{x | n \in E_x\}$ è saturo, perché $B_n = \{x | x \in \mathcal{B}_n\}$ con $\mathcal{B}_n = \{f \in \mathcal{C} | n \in cod(f)\}$ e:

- $B_n \neq \emptyset$ se e_1 tale che $\phi_{e_1} = id\ n \in E_{e_1} = \mathbb{N} \Rightarrow e_1 \in B_n$
- $B_n \neq \mathbb{N}$ se e_0 tale che $\phi_{e_0} = \emptyset\ n \notin E_{e_0} \Rightarrow e_0 \notin B_n$

Quindi per il teorema di Rice anche B_n non è ricorsivo.
Sia $f : \mathbb{N} \to \mathbb{N}$ una funzione $B_f = \{x | \phi_x = f\}$ è saturo per definizione e per il teorema di Rice non è ricorsivo a meno che $f \notin \mathcal{C}$ perché in quel caso si ha $B_f = \emptyset$. Infatti, se $f \in \mathcal{C}$, $B_f \neq \emptyset$ perché banalmente esiste un programma che non la calcola. $B_f \neq \mathbb{N}$ perché sia e_2 tale che $\phi_{e_2} \neq f$ ed esiste perché le funzioni calcolabili sono infinite si ha che $e_2 \notin B_f$. Quindi per Rice B_f non è ricorsivo.
$C = \{x | x \in W_x \cap E_x\}$ non sappiamo se questo insieme è saturato, ma intuitivamente se proviamo a definire $C = \{x | \phi_x \in \mathcal{A}\}$ abbiamo dei problemi a definire $\mathcal{A}$.
Mostrare che $K \leq_m C$.

TODO

$D = \{x | E_x infinito\}$ dire se è ricorsivo o meno

TODO

6.4 Insiemi ricorsivamente enumerabili (RE)

$A \subseteq \mathbb{N}$ è **RE** se la sua funzione semi-caratteristica è calcolabile:

$$SC_A(x) = \begin{cases} 1 & x \in A \\ \uparrow & x \notin A \end{cases}$$

Il fatto che un insieme A è RE equivale a dire che il predicato "$x \in A$" è semi-decidibile.

6.4.1 Ricorsivo $\subseteq$ RE

Se $A \subseteq \mathbb{N}$ è ricorsivo $\Leftrightarrow A, \bar{A}$ sono ricorsivamente enumerabili.

A ricorsivo $\Rightarrow$ A RE

Sia A ricorsivo, allora la sua funzione caratteristica è

$$\mathcal{X}_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

allora

$$SC_A = 1\Big(\mu w \overline{\text{sg}}\big(\mathcal{X}_A(x)\big)\Big)$$

è definita per minimalizzazione di una combinazione di funzione calcolabili e pertanto è anch'essa calcolabile.

Per dimostrare che anche $\bar{A}$ è ricorsivamente enumerabile basta osservare che se A è ricorsivo anche il suo complementare $\bar{A}$ è ricorsivo e quindi per quanto dimostrato finora è anche ricorsivamente enumerabile.

$A, \bar{A}$ RE $\Rightarrow A$ ricorsivo

Dato che A e $\bar{A}$ sono RE e quindi SC_A e $SC_{\bar{A}}$ sono calcolabili.

Si possono quindi trovare e_1, e_0 tali che $\phi_{e_1} = SC_A 1$ e $\phi_{e_0} = SC_{\bar{A}}$.

Intuitivamente è possibile definire un programma che esegue in parallelo entrambi i programmi sull'input x e attende che uno dei due programmi termini. Se termina prima e_1 ritorna 1, altrimenti se termina e_0 ritorna 0.

Per fare questo con una macchina URM serve un e_0 leggermente diverso, ovvero tale che $\phi_{e_0} = \overline{\text{sg}} \cdot SC_{\bar{A}}$. Così facendo la funzione ritorna 0 quando $x \in \bar{A}$.

La funzione calcolata dal programma che esegue i due programmi in parallelo può essere definita come:

$$\mathcal{X}_A = \Big(\mu w.S\Big(e_1, x, (w)_1, (w)_2\Big) \vee S\Big(e_0, x, (w)_1, (w)_2\Big)\Big)$$

e risulta essere calcolabile.

6.4.2 L'insieme K

$$K = \{x | x \in W_x\} = \{x | \phi_x(x) \downarrow\}$$

è ricorsivamente enumerabile anche se non è ricorsivo perché

$$SC_K(x) = \begin{cases} 1 x \in K \\ \uparrow \text{ altrimenti} \end{cases} = \begin{cases} 1 \phi_x(x) \downarrow \\ \uparrow \text{ altrimenti} \end{cases} = 1(\phi_x(x))$$

che può essere calcolata utilizzando la funzione universale.

Banalmente $\overline{K}$ non è ricorsivamente enumerabile perché se lo fosse, K sarebbe ricorsivo e sappiamo che non lo è.

6.4.3 Teorema di struttura dei predicati semidecidibili

Sia $P(\vec{x})$ un predicato k-ario, $P(\vec{x})$ è semidecidibile se e sole se esiste $Q(t, \vec{x})$ decidibili tale che $P(\vec{x}) \equiv \exists t Q(t, \vec{x})$.

Ovvero il predicato P è una generalizzazione di un predicato decidibile che viene calcolato su più punti.

In generale, quantificare esistenzialmente trasforma un predicato decidibile in uno semidecidibile.

Dimostrazione

Supponiamo che $P(\vec{x}) \equiv Q(t, \vec{x})$ con Q decidibile. La funzione caratteristica di P non farà altro che cercare la un t utilizzando una minimalizzazione illimitata.

$$SC_P(\vec{x}) = \begin{cases} 1 \text{ se } P(\vec{x}) \\ \uparrow \text{ altrimenti} \end{cases} = 1\big(\mu t.\text{“}Q(t, \vec{x})\text{”}\big) = 1\big(\mu t.|\mathcal{X}_Q(t, \vec{x}) - 1|\big)$$

SC_P è calcolabile perché $\mathcal{X}_Q$ è calcolabile per ipotesi e quindi $P\vec{x}$ è semidecidibile.

Viceversa, sia $P(\vec{x})$ semidecidibile, allora anche $SC_P(\vec{x})$ è calcolabile, ovvero $SC_P = \phi_e$ per qualche $e \in \mathbb{N}$.

Il predicato $P(\vec{x})$ vale se e solo se $SC_P(\vec{x}) = \phi_e(\vec{x}) = 1 \Leftrightarrow \phi_e(\vec{x}) \downarrow \Leftrightarrow \exists t.H(e, \vec{x}, t)$, dove H è la funzione che ritorna 1 se la macchina e termina in t passi sull'input $\vec{x}$ che sappiamo essere decidibile.

Indicando con $Q(t, \vec{x}) \equiv H^{(k)}(e, \vec{x}, t)$ decidibile e $P(\vec{x}) \equiv \exists t Q(t, \vec{x})$.

6.4.4 Chiusura per quantificazione esistenziale (teorema di proiezione)

La classe degli insiemi ricorsivamente enumerabili è chiusa rispetto la quantificazione esistenziale.

Dato $P(x, \vec{y}) \subseteq \mathbb{N}^{k+1}$ semidecidibile, allora anche $P'(\vec{y}) \equiv \exists x.P(x, \vec{y})$ è semidecidibile.

Dimostrazione

Per ipotesi $P(x, \vec{y})$ è semidecidibile e quindi per il teorema di struttura può essere espresso come

$$P(x, \vec{y}) = \exists t.Q(t, x, \vec{y})$$

con Q decidibile e utilizzando ciò è possibile definire:

$$P'(x, \vec{y}) \equiv \exists x.\exists t.Q(t, x, \vec{y}) \equiv \exists w.\underbrace{Q((w)_1, (w)_2, \vec{y})}_{Q'}$$

con $Q'(w, \vec{y}) \equiv Q((w)_1, (w)_2, \vec{y})$, che ha come funzione caratteristica $\mathcal{X}_{Q'} = \mathcal{X}_Q((w)_1, (w)_2, \vec{y})$ calcolabile.

Si ha quindi che Q' è decidibile e quindi per il teorema di struttura $P'(\vec{y})$ è semidecidibile.

6.4.5 Chiusura per and e or

Siano $P_1(\vec{x})$ e $P_2(\vec{x})$ semidecidibili, allora:

1. $P_1(\vec{x}) \vee P_2(\vec{x})$
2. $P_1(\vec{x}) \wedge P_2(\vec{x})$

sono semidecidibile.

Dimostrazione

Per il teorema di struttura possiamo vedere i due predicati come la quantificazione esistenziale di due predicati Q_1 e Q_2 decidibili.

Si ha quindi che i due predicati possono essere visti come

1. $P_1(\vec{x}) \vee P_2(\vec{x}) = \exists t Q_1(t, \vec{x}) \vee \exists t Q_2(t, \vec{x}) = \exists t(Q_1(t, \vec{x}) \vee Q_2(t, \vec{x}))$
2. $P_1(\vec{x}) \wedge P_2(\vec{x}) = \exists t Q_1(t, \vec{x}) \wedge \exists t Q_2(t, \vec{x}) = \exists w(Q_1((w)_1, \vec{x}) \wedge Q_2((w)_2, \vec{x}))$

Siccome i predicati Q_1 e Q_2 sono decidibili, anche le loro congiunzioni/disgiunzioni sono decidibili e quindi, sempre per il teorema di struttura, la congiunzione e la disgiunzione di predicati semidecidibili è semidecidibile.

6.4.6 La quantificazione universale

La quantificazione universale è rognosa da trattare perché anche un predicato decidibile, se viene quantificato universalmente può diventare non semidecidibile.

Intuitivamente questo è vero perché risulta indefinito andare a testare un predicato su infiniti valori.

Ad esempio $Q(t, x) \equiv \neg H(x, x, t)$ è decidibile, ma $Q'(x) \equiv \forall t.Q(t, x) \equiv \phi_x(x) \uparrow \equiv x \in \overline{K}$ che non è neanche semidecidibile.

Si ha anche che $x \in K$ è semidecidibile ma $\neg(x \in K) \equiv x \in \overline{K}$ non semidecidibile.

6.4.7 Riducibilità per gli insiemi RE

Dati due insiemi $A, B \subseteq \mathbb{N}$ e $A \leq_m B$

1. se B è RE $\Rightarrow$ A è RE
2. se A non è RE $\Rightarrow$ B non è RE

Questo può essere utilizzato per dimostrare che A è ricorsivamente enumerabile, con la riduzione $A \leq_m K$ oppure che A non è ricorsivamente enumerabile $\overline{K} \leq_m A$.

Dimostrazione

Per il primo punto: sia $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale e $x \in A \Leftrightarrow f(x) \in B$ per riduzione, allora

$$SC_A(x) = SC_B(f(x))$$

è calcolabile per composizione di funzioni calcolabili e quindi A è RE.
L'altro caso è analogo.

6.4.8 Perché RE? (Teorema di etimologia)

Da dove deriva il nome ricorsivamente enumerabili?

L'enumerabile deriva dal fatto che esiste una funzione $f : \mathbb{N} \to A$ suriettiva che equivale a dire che $f : \mathbb{N} \to \mathbb{N}$, totale e tale che $cod(f) = A$.

Il ricorsivo invece richiama il fatto che l'enumerazione può essere fatta con una funzione calcolabile.

Si ha quindi che l'insieme degli insiemi ricorsivamente enumerabili è effettivamente ricorsivamente enumerabile.

Dimostrazione

Sia $A \subseteq \mathbb{N}$, A è RE se e solo se $A = \emptyset$ oppure esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile, totale e tale che $A = cod(f)$.

$\Rightarrow$ Sia A RE.

Se $A = \emptyset$ il teorema è trivialmente vero.

Se $A \neq \emptyset$, sia $a_0 \in A$ fissato e sia e un programma tale che $\phi_e = SC_A$.

Il programma e può essere modificato in modo che possa eseguire al massimo t passi e che ritorni x se $x \in A$ e riesce a deciderlo in massimo t passi, altrimenti ritorna a_0. Si ha quindi che $\phi_{e'}$ ha come codominio A ed è totale.

Più formalmente

$$f(w) = \begin{cases} (w)_1 \text{ se } H(e,(w)_1,(w)_2) \\ a_0 \text{ altrimenti} \end{cases} \quad = (w)_1 \mathcal{X}_H(e,(w)_1,(w)_2) + a_0 \mid 1 - \mathcal{X}_H(e,(w)_1,(w)_2) \mid$$

f è totale perché è sempre definita ed è calcolabile perché è la composizione di funzioni calcolabili.

Inoltre, $A = cod(f)$ perché se $x \in A \Rightarrow \phi_e(x) = SC_A(x) = 1$ e quindi esiste un numero finito di passi t entro i quali l'esecuzione di e termina. Sia $w \in \mathbb{N}$ tale che $(w)_1 = x$ e $(w)_2 = t$, si ha che $f(w) = x$ e che $x \in cod(f)$. w esiste e ce ne sono possibilmente infiniti, perché basta un numero naturale tale che $w = 2^x \cdot 3^t \cdot \ldots$

Viceversa se $z \in cod(f)$, $z = f(w)$ per un qualche $w \in \mathbb{N}$. Ci sono quindi due possibilità:

1. $f(w) = a_0 \in A$
2. $f(w) = (w)_1 \rightsquigarrow H(e,(w)_1,(w)_2) \Rightarrow \phi_e((w)_1) \downarrow \Rightarrow SC_A((w)_1) = 1$ e quindi $z = (w)_1 \in A$.

$\Leftarrow$ Se $A = \emptyset$, A è ricorsivo e quindi è anche RE.

Se $A = cod(f)$ con f calcolabile e totale.

$$SC_A(x) = 1\big(\mu w.|f(w) - x|\big)$$

definita per composizione di funzioni calcolabili e quindi è calcolabile il che implica che A è RE.

6.5 Teorema di Rice-Shapiro

Le proprietà del comportamento (funzione calcolata) dei programmi possono essere semidecidibili se sono sono finitarie ovvero dipendono da un numero finito di argomenti.

Ad esempio "*la funzione termina su 5*" oppure "*la funzione termina in un punto*" sono semidecidibili.

Funzione finita: $\theta : \mathbb{N} \to \mathbb{N}$ tale che $dom(\theta)$ è finito, ovvero è una funzione definita solamente su un insieme finito di elementi.

Pezzo di una funzione: date due funzioni $f, g : \mathbb{N} \to \mathbb{N}$, f è un pezzo di g, $f \subseteq g$ se $\forall x. f(x) \downarrow \Rightarrow g(x) \downarrow$ e $f(x) = g(x)$.

Più formalmente:

Sia $\mathcal{A} \subseteq \mathcal{C}$ e sia $A = \{x | \phi_x \in \mathcal{A}\}$.

Se A è ricorsivamente enumerabile, allora $\forall f \in \mathcal{C}. \Big(f \in \mathcal{A} \Leftrightarrow \exists \theta \subseteq f \in \mathcal{A}\Big)$.

6.5.1 Dimostrazione

La dimostrazione è qualcosa del tipo

1. $\exists f\ f \notin \mathcal{A}$ e $\exists \theta \subseteq f\ \theta \in \mathcal{A} \Rightarrow A$ non è RE
2. $\exists f\ f \in \mathcal{A}$ e $\forall \theta \subseteq f\ \theta \notin \mathcal{A} \Rightarrow A$ non è RE

Dimostrazione caso 1

$$\exists f\ f \notin \mathcal{A} \text{ e } \exists \theta \subseteq f\ \theta \in \mathcal{A} \Rightarrow A \text{ non è RE}$$

Sia $f \notin \mathcal{A}$ e sia $\theta \subseteq f\ \theta \in \mathcal{A}$.

Per provare che A non è RE possiamo provare a fare la riduzione $\overline{K} \leq_m A$.

Definiamo quindi

$$\begin{aligned} g(x,y) &= \begin{cases} \theta(y)x \in \overline{K} \\ f(y)x \in K \end{cases} \\ &= \begin{cases} \uparrow\ x \in \overline{K} \text{ e } y \notin dom(\theta) \\ f(y)x \in \overline{k} \text{ e } y \in dom(\theta) \\ f(y)x \in K \end{cases} \\ &= \begin{cases} f(y)x \in K \text{ oppure } y \in dom(\theta) \\ \uparrow \text{ altrimenti} \end{cases} \end{aligned}$$

$x \in K$ è semidecidibile, quindi il predicato $Q(x,y) \equiv$ "$x \in K$" $\vee$ "$y \in dom(\theta)$" è semidecidibile.

Si può quindi definire g come:

$$g(x,y) = f(y) \cdot SC_Q(x,y)$$

e dato che sia la funzione caratteristica, sia f sono calcolabili, anche g è calcolabile.

Essendo calcolabile, per il teorema SMN $g(x,y) = \phi_{S(x)}(y)$ per $S : \mathbb{N} \to \mathbb{N}$. S è quindi una funzione di riduzione per $\overline{K} \leq_m A$:

- $x \in \overline{K}$: $\phi_{S(x)}(y) = g(x,y) = \begin{cases} f(y)x \in K \text{ oppure } y \in dom(\theta) \\ \uparrow \text{ altrimenti} \end{cases} = \theta(y)\forall y \Rightarrow \phi_{S(x)} = \theta$ ed essendo A saturo, $S(x) \in A$
- $x \in K$: $\phi_{S(x)}(y) = g(x,y) = f(y)\forall y \Rightarrow \phi_{S(x)} = f \notin \mathcal{A} \Rightarrow S(x) \notin A$

Si ha quindi che $\overline{K} \leq_m A$ e quindi A non è ricorsivamente enumerabile.

Dimostrazione caso 2

$$\exists f\ f \in \mathcal{A} \text{ e } \forall \theta \subseteq f\ \theta \notin \mathcal{A} \Rightarrow A \text{ non è RE}$$

Anche in questo caso dimostriamo che $\overline{K} \leq_m A$.

$$g(x,y) = \begin{cases} f(y) \text{ se } \neg H(x,x,y) \\ \uparrow \text{ altrimenti} \end{cases}$$
$$= f(y) \cdot 1(\mu w.\mathcal{X}_H(x,x,y))$$

g è calcolabile e quindi per il teorema SMN esiste $S : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $\phi_{S(x)}(y) = g(x,y)$.

S è la funzione di riduzione di $\overline{K} \leq_m A$:

- $x \in \overline{K}$: $\forall y \neg H(x,x,y) \Rightarrow \phi_{S(x)}(y) = f(y) \Rightarrow \phi_{S(x)} = f \in \mathcal{A} \Rightarrow S(x) \in A$
- $x \in K$: $\exists y_0$ tale che $H(x,x,y_0)$ prendo il minimo y_0 tale che $\neg H(x,x,y)\forall y < y_0$ e $H(x,x,y)\forall y \geq y_0$. Si ha quindi che $\phi_{S(x)}(y) = g(x,y) = \begin{cases} f(y)y < y_0 \\ \uparrow \text{ altrimenti} \end{cases}$. Essendo $\phi_{S(x)} \subseteq f$ ed è finito $(dom(\phi_{S(x)}) \subseteq [0, y_0[)$ e quindi $\phi_{S(x)} \notin \mathcal{A} \Rightarrow S(x) \notin A$.

6.5.2 Applicazioni del teorema

$A = \{x|\phi_x \text{ totale}\}$ e $\mathcal{A} = \{f|f\text{totale}\}$.

La funzione $id \in \mathcal{A}$ e per ogni $\theta \subseteq id, \theta$ finita , θ non è totale, quindi $\theta \notin \mathcal{A}$. Per il teorema di Rice Shapiro A non è RE.

Per dimostrare che $\overline{A}$ non è RE, si può considerare $\mathcal{A} = \{f|f \text{ non totale}\}$, utilizzando RS si riesce a trovare $\emptyset \in \overline{A}$ e $\emptyset \subseteq id$ con $id \notin \overline{\mathcal{A}}$.

$U = \{x|\phi_x = 1\}$ e $\mathcal{U} = \{1\}$.

$1 \in \mathcal{U}$ e $\forall \theta \subseteq 1\ \theta \notin \mathcal{U}$, quindi U non è RE per Rice Shapiro.

Ma non vale nel caso generale $\forall f\ A_f = \{x|\phi_x = f\}$, $\mathcal{A}_f = \{f\}$.

Se $f \notin \mathcal{C}$, $A_f = \emptyset$ e quindi A_f è ricorsivo.

Se invece $f \in \mathcal{C}$, bisogna distinguere i vari casi.

- $f = \emptyset$: $\mathcal{A}_f = \{x|\phi_x = \emptyset\} \neq \emptyset \neq \mathbb{N}$ e quindi A_f non è ricorsivo per il teorema di Rice. $\overline{A}_f = \{x|\phi_x \neq \emptyset\}$ è RE perché la sua funzione caratteristica è $SC_{\overline{A}_f}(x) = 1\Big(\mu w.H(x,(w)_1,(w)_2)\Big)$. Così facendo se $x \in \overline{A}_f$ allora $W_x \neq \emptyset$, sia $y \in W_x$ ovvero $\phi_x(y) \downarrow \Rightarrow H(x,y,t)$ per qualche t, segue che $\mu w.H(x,(w)_1,(w)_2) \downarrow$ e quindi la funzione caratteristica è corretta. Il tutto vale anche in $\Leftrightarrow$. Ricapitolando A_f non è ricorsivo, $\overline{A}_f$ è ricorsivamente enumerabile e pertanto A_f non è RE, perché altrimenti A_f sarebbe ricorsivo.
- $f \neq \emptyset$ e finita: A_f non è RE, $f \in \mathcal{A}_f = \{f\}$ considero $g(x) = \begin{cases} f(x)x \in dom(f) \\ 1 \text{ altrimenti} \end{cases}$, $g \notin \mathcal{A}_f$ e quindi per Rice Shapiro A_f non è RE.
- f non finita: $f \in \mathcal{A}_f = \{f\}$ ma $\forall \theta \subseteq f\ \theta \notin A_f$ e quindi sempre per Rice Shapiro A_f non è RE.

In entrambe i casi $\emptyset \in \overline{A}_f$ e $\emptyset \subseteq f$ e $f \notin \overline{A}_f$ e quindi per Rice Shapiro $\overline{A}_f$ non è RE.
Ricapitolando:

- $f \notin \mathcal{C}$: A_f e $\overline{A}_f$ sono ricorsivi $(\emptyset, \mathbb{N})$
- $f \in \mathcal{C}$
  - A_f non è RE.
  - $\overline{A}_f$ è RE se $f = \emptyset$, non RE altrimenti.

6.5.3 Rice Shapiro non è un se e solo se

$\mathcal{A} \subseteq \mathcal{C}$, $A = \{x | \phi_x \in \mathcal{A}\}$.
Sappiamo che se A è RE allora $\forall f \big(f \in \mathcal{A} \Leftrightarrow \exists \theta \subseteq f \ \theta \in \mathcal{A}\big)$ ma l'altro verso non è sempre vero.
$\mathcal{A} = \{f \in \mathcal{C} | dom(f) \cap \overline{K} = \emptyset\}$.
Se $f \in \mathcal{A} \Rightarrow dom(f) \cap \overline{K} \neq \emptyset$, sia $x_0 \in dom(f) \cap \overline{K}$, allora $\theta(x_0) = \begin{cases} f(x) \text{ se } x = x_0 \\ \uparrow \text{ altrimenti} \end{cases}$ Per costruzione $\theta \subseteq f$ e $x_0 \in dom(f) \cap \overline{K} \neq \emptyset$, quindi $\theta \in \mathcal{A}$.
Viceversa se $\theta \subseteq f \, \theta \in \mathcal{A}$, $dom(f) \cap \overline{K} \neq \emptyset$ e $dom(\theta) \subseteq dom(f)$ e quindi $dom(\theta) \cap \overline{K} \subseteq dom(f) \cap \overline{K}$ e quindi $dom(f) \cap \overline{K} \neq \emptyset$ e $f \in \mathcal{A}$.
Abbiamo quindi dimostrato che la parte è unitaria, adesso osserviamo che A non è RE.
Dato x_0, $\theta(x) = \begin{cases} 1 \text{ se } x = x_0 \\ \uparrow \text{ altrimenti} \end{cases}$, $\theta \in \mathcal{A} \Leftrightarrow x_0 \in \overline{K}$.
Questo può essere formalizzato considerando $g(x, y) = \begin{cases} 1 \text{ se } x = y \\ \uparrow \text{ altrimenti} \end{cases}$ che è calcolabile con $1\big(\mu z.|x - y|\big)$ e quindi per il teorema SMN esiste $S : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che $g(x, y) = \phi_{S(x)}(y)$ e che sia funzione di riduzione per $\overline{K} \leq_m A$, perché $dom(\phi_{S(x)}) = \{x\}$ e quindi se $x \in \overline{K} \Rightarrow dom(\phi_{S(x)}) \cap \overline{K} \neq \emptyset \Rightarrow S(x) \in A$. Analogamente se $x \notin \overline{K}$ $dom(\phi_{S(x)}) \cap \overline{K} = \emptyset \Rightarrow x \notin A$.
Abbiamo quindi un contro esempio che mostra che A non è RE, anche se valgono le condizioni di Rice Shapiro.

6.6 Primo teorema di ricorsione

Cosa vuol dire essere calcolabile nel lavorare con le funzioni?
Funzionale: una qualunque funzione totale del tipo $\Phi : \mathcal{F}(\mathbb{N}^k) \to \mathcal{F}(\mathbb{N}^h)$ dove $\mathcal{F}(\mathbb{N}^k) = \{f | f : \mathbb{N}^k \to \mathbb{N}\}$.
Quando è che un funzionale è ricorsivo? (calcolabile).
Tipicamente a noi interessa sapere il valore di $\Phi(f)(x)$ per qualche x ma non infiniti x. Questo è calcolabile quanto il risultato dipende da una parte finita di f: $\vartheta \subseteq f$.
Si ha quindi che $\Phi(f)(x)$ è **ricorsivo** se esiste ϕ calcolabile tale che

$$\Phi(f)(x) = \phi(\vartheta, x) \text{ per qualche } \vartheta \subseteq f \text{ finita}$$

Funzioni finite come numeri

$$\hat{\vartheta} = \begin{cases} 0 \text{ se } \vartheta = \emptyset \\ \prod\limits_{x \in dom(\vartheta)} P_x^{\vartheta(x)+1} \end{cases}$$

Così facendo, dato un numero $z \in \mathbb{N}$ tale che $z = \hat{\varphi}$ è possibile determinare se $x \in dom(\vartheta)$ andando a verificare se $P_x | z$.
In modo simile è possibile applicare la parte di funzione rappresentata da z ad un numero x.

$$app(z,x) = \begin{cases} (z)_x \dot{-} 1 \text{ se } x \in dom(z) \\ \uparrow \text{ altrimenti} \end{cases}$$

Funzionare ricorsivo

Un funzionale

$$\Phi : \mathcal{F}(\mathbb{N}^k) \to \mathcal{F}(\mathbb{N}^h)$$

si dice **ricorsivo** se esiste una funzione $\phi : \mathbb{N}^{h+1} \to \mathbb{N}$ tale che per ogni $f \in \mathcal{F}(\mathbb{N}^k)$ e per ogni $\vec{x} \in \mathbb{N}^h, y \in \mathbb{N}$ abbiamo

$$\Phi(f)(\vec{x}) = y \text{ sse } \exists \vartheta \subseteq f \text{ finito } \phi(\vartheta, \vec{x}) = y$$

Ad esempio, il funzionale $\Phi : \mathcal{F}(\mathbb{N}^1) \to \mathcal{F}(\mathbb{N}^1)$:

$$\Phi(f)(x) = f(x) + 1$$

è ricorsivo perché posso trovare

$$\begin{aligned} \phi(\hat{\vartheta}) &= \hat{\vartheta}(x) + 1 \\ &= app(\hat{\vartheta}, x) + 1 \end{aligned}$$

Anche la funzione di Ackermann è un funzionale ricorsivo $\Psi : \mathcal{F}(\mathbb{N}^2) \to \mathcal{F}(\mathbb{N}^2)$.

6.6.1 Calcolabilità dei funzionali ricorsivi

Se prendo un funzionale ricorsivo $\Phi : \mathcal{F}(\mathbb{N}^k) \to \mathcal{F}(\mathbb{N}^h)$ e prendo una funzione calcolabile $f : \mathbb{N}^k \to \mathbb{N}$, allora anche l'immagine $\Phi(f) : \mathbb{N}^h \to \mathbb{N}$ è calcolabile.

6.6.2 Myhill - Shepherodson

$$\Phi \text{ ricorisvo e } f \text{ calcolabile } \Rightarrow \Phi(f) \text{ calcolabile}$$

$$\Phi(\phi_e) \text{ calcolabile } = \phi_{e'} = \phi_{h(e)}$$

Diciamo che $h : \mathbb{N} \to \mathbb{N}$ è **estensionale** se

$$\forall e, e' \; \phi_e = \phi_{e'} \to \phi_{h(e)} = \phi_{h(e')}$$

Il **Teorema di Myhill - Shepherdson - Parte 1** è quindi il seguente:
Sia $\Phi : \mathcal{F}(\mathbb{N}^k) \to \mathcal{F}(\mathbb{N}^h)$ un funzionale ricorsivo. Allora esiste una funzione calcolabile, totale ed estensionale $h : \mathbb{N} \to \mathbb{N}$ tale che $\forall e \in \mathbb{N}$:

$$\Phi(\phi_e) = \phi_{h(e)}$$

La **Parte 2** del teorema dice anche che:
Sia $h : \mathbb{N} \to \mathbb{N}$ una funzione calcolabile, totale ed estensionale. Allora esiste un'unico funzionare ricorsivo $\Phi : \mathcal{F}(\mathbb{N}^k) \to \mathcal{F}(\mathbb{N}^h)$ tale che

$$\Phi(\phi_e^{(k)}) = \phi_{h(e)}^h$$

6.6.3 Il primo teorema di ricorsione (Kleene)

Sia $\Phi : \mathcal{F}(\mathbb{N}^k) \to \mathcal{F}(\mathbb{N}^k)$ un funzionale ricorsivo. Allora questo funzionale Φ ha un minimo punto fisso $f_\Phi : \mathbb{N}^k \to \mathbb{N}^k$ calcolabile.

1. $\Phi(f_\phi) = f_\phi$
2. $\Phi(g) = g \Rightarrow f_\phi \subseteq g$
3. f_Φ è calcolabile.

Inoltre

$$f_\phi = \bigcup_{n \in \mathbb{N}} \Phi^n(\emptyset)$$

Ad esempio, la funzione $x \dot{-} 1$ può essere scritta come

$$\Phi(f)(x) = \begin{cases} 0 \text{ se } x = 0, 1 \\ f(x-1) + 1 \end{cases}$$

si ha quindi che per il primo teorema di ricorsione:

$$\begin{aligned} \Phi^n(\emptyset)(3) &= \Phi^{n-1}(\emptyset)(2) + 1 \\ &= (\Phi^{n-2}(\emptyset)(1) + 1) + 1 \\ &= 0 + 1 + 1 = 2 \end{aligned}$$

segue al dimostrazione del perché la ricorsione primitiva è calcolabile

Ad esempio la minimalizzazione illimitata $\mu y.f(\vec{x}, y)$ può essere definita dal funzionale:

$$\Phi(h)(\vec{x}, y) = \begin{cases} y \text{ se } h(\vec{x}, y) = 0 \\ h(\vec{x}, y+1) \text{ se } h(\vec{x}, y) \downarrow \neq 0 \\ \uparrow \text{ altrimenti} \end{cases}$$

Si ha quindi che:

$$\mu y.f(\vec{x}, y) = f_{\Phi_\mu}(\vec{x}, 0)$$

La cosa importante è che il funzionale ricorsivo sia definito utilizzando solamente un numero finito di volte la funzione che riceve come argomento.

Considerando il funzionale

$$\Phi(f)(x) = \begin{cases} 1 \text{ se } x = 0 \\ f(x+1) \text{ se } x > 0 \end{cases}$$

Si ottengono come possibili punti fissi:

$$f_\Phi(x) = \begin{cases} 1 \text{ se } x = 0 \\ \uparrow \text{ altrimenti} \end{cases} \qquad \text{e} \qquad f_\Phi(x) = \begin{cases} 1 \text{ se } x = 0 \\ k \text{ se } x > 0 \end{cases}$$

6.7 Secondo teorema di ricorsione

Sia $h : \mathbb{N} \to \mathbb{N}$ calcolabile e totale.

Allora esiste $n \in \mathbb{N}$ tale che $\phi_{h(n)} = \phi_n$. Ovvero esiste un programma che anche se viene modificato dalla funzione h, continua a calcolare la stessa funzione.

La differenza con il primo teorema è che viene persa l'ipotesi dell'estensionabilità.

6.7.1 Dimostrazione

$$\begin{aligned} g(x,y) &= \phi_{h(\phi_x(x))}(y) \\ &= \Psi_U\big(h(\Psi_U(x,x)), y\big) \end{aligned}$$

Dal momento che la funzione universale è calcolabile, anche g è calcolabile, e quindi per il teorema SMN esiste $S : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che

$$g(x,y) = \phi_{S(x)}(y)$$

Essendo S calcolabile, esiste un programma $e \in \mathbb{N}$ tale che $\phi_e = S$ e quindi:

$$g(x,y) = \phi_{\phi_e(x)}(y)$$

Andando a considerare il caso in cui $x = e$ si ha:

$$\phi_{h(\phi_e(e))}(y) = \phi_{\phi_e(e)}(y)$$

detto $n = \phi_e(e)$ si ottiene che

$$\phi_{h(n)}(y) = \phi_n(y)\ \forall y$$

che è proprio quello che volevamo.

Inoltre, abbiamo la garanzia che $\phi_e(e) \neq\uparrow$ perché S è totale.

La dimostrazione fatta è una diagonalizzazione, perché h può essere vista come una enumerazione di funzioni calcolabili.

6.7.2 Conseguenza: Teorema di Rice

$A \neq \emptyset, \mathbb{N}$ saturato. Allora A non è ricorsivo.

Dimostrazione utilizzando il secondo teorema di ricorsione

Supponiamo per assurdo che A sia ricorsivo. Siano quindi $a_1 \in A$ e $a_0 \notin A$, risulta possibile definire

$$\begin{aligned} f(x) &= \begin{cases} a_0 \text{ se } x \in A \\ a_1 \text{ se } x \notin A \end{cases} \\ &= a_0\chi_A(x) + a_1\chi_{\overline{A}}(x) \end{aligned}$$

calcolabile e totale.

Per il secondo teorema di ricorsione esiste un n tale che $\phi_n = \phi_{f(n)}$.

Se $n \in A$ allora $f(n) = a_0 \notin A$ ma questo non può essere, perché $\phi_n \neq \phi_{f(n)}$ a causa del fatto che A è saturo e che che $f(n) \notin A$.

Allo stesso modo se $n \notin A$, allora $f(n) = a_1 \in A$ e quindi $\phi_n \neq \phi_{f(n)}$.

Segue che A non è ricorsivo.

Capitolo 7

Altri esercizi

7.1 Teoremi di ricorsione

7.1.1 Es 1

Dimostrare che K non è ricorsivo.

Soluzione

Assumiamo che K sia ricorsivo, sia $e_0 \in \overline{K}$ e tale che $\phi_{e_0} = \emptyset$.

Dato un x, se $x \in K$ allora cerchiamo una funzione che lo manda in e_0 e se $x \in \overline{K}$ lo mandiamo in $e_1 \in K$ tale che $\phi_{e_1} = id$.

La nostra funzione sarà quindi

$$f(x) = e_0 \cdot \mathcal{X}_K(x) + e_1 \cdot \mathcal{X}_{\overline{K}}(x)$$

risulta essere totale, perché definita su ogni input e calcolabile, perché entrambe le funzioni caratteristiche sono calcolabili (**perché c'è l'assunzione che *K* sia ricorsivo**).

Per il secondo teorema di ricorsione esiste un punto fisso, $\exists e \ \phi_{f(e)} = \phi_e$. Per questo programma abbiamo due possibilità

- $e \in K$, ma quindi $\phi_e(e) \downarrow$ e siccome $e \in K$, $f(e) = e_0$ e quindi $\phi_{f(e)}(e) = \phi_{e_0}(e) \uparrow$. Si ha quindi che $\phi_{e_0}(e) \neq \phi_e(e)$ che contraddice l'ipotesi.
- $e \notin K$, ma quindi $\phi_e(e) \uparrow$ e per definizione $f(e) = e_1$ e quindi $\phi_{f(e)} = \phi_{e_1}(e) = \downarrow$. Si ha quindi che $\phi_e \neq \phi_{f(e)}$ che contraddice l'ipotesi.

Da questo segue che K non è ricorsivo, perché viene contraddetta l'assunzione che lo sia.

7.1.2 Esercizio 2

K non è saturo.

Soluzione

Abbiamo già osservato che per sapere che K non è saturo, basta avere un programma P che riceve in input dei programmi espressi come numeri e controlla se riceve in input se stesso. Se la risposta è si, ritorna 1, altrimenti $\uparrow$.

Assumendo che questo programma si possa definire, è possibile definire anche un altro programma, aggiungendo un'istruzione che non fa niente, in modo da cambiare rappresentazione numerica. Questo programma continua a terminare solo se riceve in input la versione originale del programma.

Più formalmente

$$\phi_{e_1}(x) = \begin{cases} 1 & x = e_1 \\ \uparrow & \text{altrimenti} \end{cases}$$

SI ha che $e_1 \in K$ perché $\phi_{e_1}(e_1) = 1 \downarrow$ e siccome ci sono infiniti indici per calcolare ϕ_{e_1}, si può trovare $e_2 \neq e_1$ tale che $\phi_{e_2} = \phi_{e_1}$.

Si ha che $e_2 \notin K$ dato che $\phi_{e_2}(e_2) = \phi_{e_1}(e_2) \uparrow$, segue quindi che K non è saturo.

Resta da dimostrare che esiste e_1.

Definiamo

$$g(e, x) = \begin{cases} 1 & x = e \\ \uparrow & \text{altrimenti} \end{cases} = 1(\mu z.|e - x|)$$

Questa funzione è calcolabile per minimalizzazione illimitata e quindi per il teorema SMN esiste una funzione $S : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che $\phi_{S(e)}(x) = g(e, x)$.

Siccome S è calcolabile e totale, per il secondo teorema di ricorsione esiste e_1 tale che $\phi_{S(e_1)} = \phi_{e_1}$. Ovvero:

$$\phi_{e_1}(x) = \phi_{S(e_1)}(x) = g(e_1, x) = \begin{cases} 1 & x = e_1 \\ \uparrow & \text{altrimenti} \end{cases}$$

e_1 è quindi il programma che cercavamo.

7.2 Esercizio 3 di un vecchio appello

Studiare la ricorsività dell'insieme $A = \{x \in \mathbb{N} | \mathbb{P} \subseteq W_x\}$.

7.2.1 Soluzione

Si può osservare che l'insieme è saturo, perché riguarda la proprietà di una funzione. Trattandosi di un insieme probabilmente saturo è impossibile che sia ricorsivo, inoltre dal momento che per verificare che il dominio contenga solo numeri pari è necessario andare a controllare tutti i possibili valori di input e quindi è probabile che non sia neanche RE.

Per quanto riguarda $\overline{A} = \{x \in \mathbb{N} \mid \mathbb{P} \nsubseteq \mathbb{N}\}$, il ragionamento è simile perché per sapere se un numero pari non compare nel dominio di una funzione richiede di provare tutti i valori e quindi probabilmente anche questo insieme non è RE.

Come prima cosa, dimostriamo che A è saturo.

$$A = \{x \in \mathbb{N} \mid \phi_x \in \mathcal{A}\}$$
$$\mathcal{A} = \{f \in \mathcal{C} \mid \mathbb{P} \in dom(f)\}$$

Per Rice Shapiro A non è RE perché $id \in A$, ma per ogni $\vartheta \subseteq f$ finita, dato che $dom(\vartheta)$ è finito si ha che $\mathbb{P} \nsubseteq dom(\vartheta)$ e quindi $\vartheta \notin \mathcal{A}$.

Anche $\overline{A}$ non è RE per Rice Shapiro, $id \notin \overline{\mathcal{A}}$ e $\vartheta = \emptyset \in \overline{\mathcal{A}}$, $\vartheta \subseteq id$ e parte finita, quindi per Rice Shapiro l'insieme non è RE.

7.2.2 Soluzione per riduzione

$\overline{K} \leq_m A$, ovvero devo trovare una funzione f calcolabile e totale che manda gli elementi di $\overline{K}$ in A e gli elementi di K in $\overline{A}$.

$$x \in \overline{K} \rightsquigarrow \phi_{f(x)} \text{ totale}$$
$$x \in K \rightsquigarrow \phi_{f(x)} \text{ finita}$$

Serve quindi una funzione

$$g(x,y) = \begin{cases} \text{totale se } x \in \overline{K} \\ \text{finita se } x \in K \end{cases} = \begin{cases} 0 \text{ se } \neg H(x,x,y) \\ \uparrow \text{ se } H(x,x,y) \end{cases} = \mu z.\mathcal{X}_H(x,x,y)$$

La funzione è calcolabile è varrà sempre 0 se la computazione del programma x termina su input x in y passi e questo è sempre vero se $x \in \overline{K}$.

Per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $\phi_{f(x)} = g(x,y)$.

Questa funzione f è funzione di riduzione da $\overline{K} \leq_m A$ perché

- $x \in \overline{K}$: si ha che $\phi_x(x) \uparrow$ e quindi il predicato $H(x,x,y)$ è sempre falso. Per cui la funzione $g(x,y)$ è sempre uguale a 0 per ogni y e in particolare $\mathbb{P} \subseteq W_{f(x)} = \mathbb{N}$ e quindi $f(x) \in A$.
- $x \in K$: si ha che $\phi_x(x) \downarrow$ e quindi $\exists y_0$ tale che $\forall y > y_0$ vale $H(x,x,y)$ e quindi da quel punto in poi $\phi_{f(x)}(y) = g(x,y) = \uparrow \forall y > y_0$ e quindi $W_{f(x)} = dom(\phi_{f(x)}) \subseteq [0, y_0]$ finito e quindi $\mathbb{P} \nsubseteq W_{f(x)}$ e pertanto $f(x) \notin A$.

7.3 Altro esercizio sulla ricorsività

$A = \{x \in \mathbb{N} \mid \exists k \,.\, \phi_x(x+3k) \uparrow\}$, studiare la ricorsività.

7.3.1 Soluzione

Probabilmente questo insieme non è RE perché ci si chiede se la funzione non è definita per certi valori.

Per quanto riguarda $A = \{x \mid \forall k \, \phi_x(x+3k) \downarrow\}$, anche in questo caso probabilmente non è RE, perché l'insieme è quantificato universalmente.

Tra l'altro l'insieme probabilmente non è saturo perché non riguarda una caratteristica della funzione.

Procediamo quindi per riduzione $\overline{K} \leq_m A$, cercando di mandare tutti gli $x \in \overline{K}$ in un elemento di A, in particolare nella funzione $\emptyset$. Mentre se $x \in K$, $f(x)$ deve finire in $\overline{A}$, ma in $\overline{A}$ ci sono tutte le funzioni totali e quindi me ne va bene una qualsiasi di queste.

Si può quindi definire

$$g(x,y) = \begin{cases} \uparrow & x \in \overline{K} \\ 1 & x \in K \end{cases} = SC_K(x)$$

e sappiamo essere calcolabile, quindi per SMN esiste la funzione calcolabile e totale $S : \mathbb{N} \to \mathbb{N}$ tale che $g(x,y) = \phi_{S(x)}(y)$.

S è funzione di riduzione di $\overline{K}$ a A:

- $x \in \overline{K}$: $\phi_{S(x)}(y) \uparrow \forall y$ e in particolare per $k = 0$ abbiamo $\phi_{S(x)}(S(x)+3k) \uparrow$ e quindi $S(x) \in A$.
- $x \in K$: $\phi_{S(x)}(y) = 1 \forall y$, allora $\forall k \phi_{S()}(S(x)+3k) = 1 \downarrow$ e quindi $S(x) \notin A$.

Quindi dato che A si riduce a $\overline{K}$, A non è RE.

Resta da studiare $\overline{A}$. Riduciamo $\overline{K} \leq_M \overline{A}$.

In questo caso, quando $x \in \overline{A}$, $f(x)$ deve essere una funzione totale e quando $x \in K$, $f(x)$ deve essere una funzione finita[1].

Si può quindi definire

$$g(x,y) = \begin{cases} 0 \text{ se } \neg H(x,x,y) \\ \uparrow \text{ se } H(x,x,y) \end{cases} = \phi_{S(x)}(y)$$

È calcolabile e quindi per SMN esiste la classica S che funziona da funzione di riduzione perché

[1] $\overline{A}$ contiene tutte le funzioni totali, A contiene tutte le funzioni finite.

- $x \in \overline{K}$: $\forall y \neq H(x,x,y)$ è sempre vero e quindi $\phi_{S(x)}(y) = 0$ e quindi $\forall k \phi_{S(x)}(S(x)+3k) = 0$ e quindi è definita. Pertanto $S(x) \in \overline{A}$.
- $x \in K$: $\phi_x(x) \downarrow$ perché $\exists y_0 > 0 . H(x,x,y)$ è vero per $\forall y \geq y_0$ e quindi $\phi_{S(x)}(y) = g(x,y) = \uparrow$ $\forall y \geq y_0$. Se $k = y_0$, $S(x)+3k \geq y_0$ e quindi la funzione $\phi_{S(x)}(S(x)+3k) \uparrow$. Quindi $S(X) \in A$.

7.4 Altro esercizio che si fa in 30 secondi (Secondo teorema di ricorsione)

Dimostrare che la funzione $\triangle : \mathbb{N} \to \mathbb{N}$ non è calcolabile.

$$\triangle(x) = min\{y | \phi_y \neq \phi_x\}$$

7.4.1 Soluzione

Il pattern è

1. Osservo che $\triangle$ è totale
2. Osservo che $\triangle$ non ha punti fissi
3. Deduco che $\triangle$ non è calcolabile perché sennò dovrebbe avere dei punti fissi

Per dimostrare che $\triangle$ è totale si può osservare che dato x esiste certamente y tale che $\phi_x \neq \phi_y$, quindi l'insieme da minimizzare non è vuoto e quindi la funzione è sempre definita.

Per dimostrare che $\triangle$ non ha punti fissi si può osservare che $\forall e \phi_{\triangle(e)} \neq \phi_e$ per definizione della funzione $\triangle$.

Segue quindi che per il secondo teorema di ricorsione $\triangle$ non è calcolabile.

7.5 Altro esercizio

Dato $X \subseteq \mathbb{N}$ indichiamo con $X+1 = \{x+1 | x \in X\}$, ovvero l'insieme dei successori.

Studiare la ricorsività di $A = \{x | E_x = W_x + 1\}$.

7.5.1 Soluzione

L'insieme è saturato perché contiene le funzioni il cui codominio è l'insieme dei successori del dominio.

Probabilmente non è RE, perché si vogliono confrontare il dominio con il codominio ed è già difficile valutare l'appartenenza di un numero ad un dominio.

Per dimostrare che non è RE usiamo Rice Shapiro, stando attenti però che la funzione $\emptyset \in \mathcal{A}$ perché $cod(\emptyset) = \emptyset$ e $dom(\emptyset) = \emptyset$. Lo stesso vale per la funzione successore S.

Si può però osservare che $id \notin \mathcal{A}$ perché $dom(id) = \mathbb{N}$ e $cod(id) = \mathbb{N} \neq \mathbb{N}+1$.

Tuttavia $\emptyset \subseteq id$ e finita, $\emptyset \in \mathcal{A}$ e quindi per Rice Shapiro A non è RE.

Per quanto riguarda il complementare di $\mathcal{A}$:

$$\overline{\mathcal{A}} = \{f | cod(f) \neq dom(f) + 1\}$$

Sappiamo che $S \notin \overline{\mathcal{A}}$ e consideriamo

$$sm(x) = \begin{cases} 2 & \text{se } x = 0 \\ 1 & \text{se } x = 1 \\ x+1 & \text{altrimenti} \end{cases}$$

Si ha quindi che $dom(sm) = \mathbb{N}$ e $cod(sm) = \mathbb{N}+1$ e quindi $sm \notin \overline{\mathcal{A}}$.

Possiamo considerare la parte finita di sm:

$$\vartheta(x) = \begin{cases} 1 & \text{se } x = 1 \\ \uparrow & \text{altrimenti} \end{cases}$$

quindi $\vartheta \subseteq sm$, $dom(\vartheta) = \{1\}$, $cod(\vartheta) = \{1\} \neq \{1\} + 1$ e quindi $\vartheta \in \overline{\mathcal{A}}$. Per Rice Shapiro si ha che $\overline{A}$ non è RE.

Capitolo 8

Riassunto in preparazione all'esame

8.1 Definizioni utili / conoscenze base

- **Teorema SMN**: Dato $m, n \geq 1$, esiste $S : \mathbb{N}^{m+1} \to \mathbb{N}$ calcolabile e totale, tale che $\phi_e^{(m+n)}(\vec{x}, \vec{y}) = \phi_{S(e,\vec{x})}^{(n)}(\vec{y})$ per qualche $\vec{x} \in \mathbb{N}^m$ e $\vec{y} \in \mathbb{N}^n$.
- **Insiemi ricorsivi e RE**: dato un sottoinsieme $X \subseteq \mathbb{N}$, se la funzione caratteristica dell'insieme è calcolabile si dice che l'insieme è **ricorsivo**, mentre se è calcolabile solo la funzione semi-caratteristica si dice che è **Ricorsivamente Enumerabile**. Se un insieme è ricorsivo, il predicato $x \in A$ è decidibile, mentre se è RE, il predicato è semi-decidibile.
- **Insieme saturo**: un insieme $A \subseteq \mathbb{N}$ si dice **saturo** se, dati $x \in A$ e $y \in \mathbb{N}$, tali che $\phi_x = \phi_y$, allora anche $y \in A$. Ovvero un insieme è saturo quando contiene tutti gli indici che calcolano una determinata funzione. Alternativamente A è saturo se e solo se esiste $\mathcal{A} \subseteq \mathcal{C}$ tale che $A = \{x \,|\, \phi_x \in \mathcal{A}\}$, cioè l'appartenenza all'insieme dipende da delle caratteristiche della funzione calcolata dal programma e non da come questo è definito.
- $K = \{x \mid \phi_x(x) \downarrow\} = \{x \mid x \in W_x\}$, ovvero l'insieme dei programmi che terminano quando ricevono se stessi in input, **non** è ricorsivo, ma è RE. L'insieme **non è saturo** (si dimostra cercando un e' che calcola la stessa funzione di e ma che non termina su se stesso).
- **Teorema di Rice**: Il teorema di Rice asserisce che qualsiasi proprietà non banale delle funzioni calcolate dai programmi non è decidibile. In altre parole, sia $A \subseteq \mathbb{N}$ l'insieme che contiene tutti i programmi la cui funzione calcolata gode di una determinata proprietà (ovvero A è un insieme saturo), se la proprietà non è banale ($A \neq \emptyset, \neq \mathbb{N}$) allora A non è ricorsivo.
- $T = \{x | \phi_x \text{ è totale}\}$, è saturo, non è ricorsivo, è RE.
- **Funzione finita**: una funzione si dice finita se è definita solamente in un numero finito di elementi.
- **Pezzo di una funzione**: date due funzioni $f, g : \mathbb{N} \to \mathbb{N}$, f è un **pezzo** di g ($f \subseteq g$) se

  $$\forall x \; f(x) \downarrow \Rightarrow g(x) \downarrow \text{ e } f(x) = g(x)$$

  Ovvero f è un pezzo di g solo se nei punti in cui è definita è uguale a g.
- $g(x, y)$: durante la **riduzione** $A \leq_m B$ si cerca una funzione $g(x, y)$ calcolabile e tale che vista come funzione di y appartenga a B solo se $x \in A$. Tipicamente viene utilizzata una funzione costante rispetto ad y, tuttavia in alcuni casi, ad esempio quando c'è la condizione $\phi_x(x) > x$ è necessario usare anche y, in modo che venga soddisfatta la condizione, ad esempio $g(x, x) > x$. Alternativamente y viene utilizzato quando la funzione deve avere determinate proprietà per qualche $n \in \mathbb{N}$.
- **Riduzioni tipiche:**
  - $K \leq_m A$ per provare che A non è ricorsivo

- $A \leq_m K$ per provare che A è RE (se so che A non è ricorsivo, posso definire SC_A per provare che è RE, senza fare la riduzione)
- $\overline{K} \leq_m A$ per provare che A non è RE.

- Una parte finitaria di una funzione totale è sempre calcolabile

8.2 Minimalizzazione illimitata

Data $f : \mathbb{N}^{k+1} \to \mathbb{N}$, si vuole definire $h : \mathbb{N}^k \to \mathbb{N}$ tale che calcoli il minimo z che azzera la funzione f, ovvero:

$$h(\vec{x}) = \mu z.f(\vec{x}, z)$$

Ci sono dei problemi se $f(\vec{x}, z)$ è sempre diversa da 0, perché in questo caso la funzione h è $\uparrow$. In modo simile c'è un problema se esiste un valore $z' < z$ per il quale la funzione f è indefinita perché, anche in questo caso, la funzione h risulta essere $\uparrow$.

Più formalmente, la minimalizzazione illimitata può essere vista come:

$$h(\vec{x}) = \mu z.f(\vec{x}, z) = \begin{cases} \text{minimo } z \text{ tale che } f(\vec{x}, z) = 0 \text{ se esiste, e } \forall z' < z, f(\vec{x}, z') \downarrow \neq 0. \\ \uparrow \text{ altrimenti} \end{cases}$$

La classe $\mathcal{C}$ delle funzioni URM calcolabili è chiusa rispetto all'operazione di minimalizzazione illimitata, ovvero, se $f : \mathbb{N}^{k+1} \to \mathbb{N}$ è in $\mathcal{C}$, allora anche $\mu z.f(\vec{x}, z)$ è in $\mathcal{C}$.

Dimostrazione: Sia P il programma URM in forma normale che calcola f. L'idea è quella di eseguire P incrementando via via un contatore, e quando viene trovato un valore che azzera la funzione, questo viene ritornato.

Si tratta quindi di scrivere un programma che esegua il programma P, ogni volta su un valore diverso e in modo che non ci siano conflitti sui registri. Come prima cosa è quindi necessario stabilire il numero di registri utilizzato da P:

$$m = \max\big(\rho(P), k\big)$$

Dopodiché è possibile definire un programma P' che:

1. Copia il contenuto dei registri $r_1, \ldots, r_k$ in $r_{m+1}, \ldots, r_{m+k}$
2. Esegue il programma P con in input il valore dei registri $r_{m+1}, \ldots, r_{m+k}$ e pone l'output del programma nel registro r_1
3. Controlla se il contenuto del registro è 0. Se è 0 ritorna z, altrimenti continua incrementando z.

```
T([1..k], [m+1..m+k])
P[m+1, ..., m+k+1 -> 1]
J(1, m+k+2, END) #LOOP
S(m+k+1)
J(1,1,LOOP)
#END
```

Dal momento che è possibile trovare un programma che calcola la minimalizzazione illimitata, questa è calcolabile.

8.3 Funzioni primitive ricorsive

La classe delle funzioni primitive ricorsive $\mathcal{PR}$ è la **minima** classe di funzioni che contiene le funzioni

(a) $zero(x) = 0$

(b) $succ(x) = x + 1$

(c) $proj(x) = U_i^k(x_1, \ldots, x_k) = x_i$

ed è chiusa rispetto a

1. composizione
2. ricorsione primitiva

Si ha che $\mathcal{PR} \subsetneq \mathcal{R} = \mathcal{C}$ perché la classe $\mathcal{R}$ delle funzioni parziali ricorsive è chiusa anche rispetto la minimalizzazione illimitata.

Da notare che la minimalizzazione illimitata permette di definire delle funzioni che non sono totali e che $\mathcal{PR} \subsetneq \mathcal{R} \cap \text{Totali}$ perché esiste la funzione di Ackermann che è totale e calcolabile ma non è primitiva ricorsiva.

8.4 Riduzione

Si hanno due problemi $\mathcal{A}$ e $\mathcal{B}$ e si vuole poter dire se $\mathcal{A}$ è più facile o più difficile di $\mathcal{B}$.

Più formalmente siano $A, B \subseteq \mathbb{N}$ gli insiemi contenenti le istanze dei problemi. Si ha che il problema $\mathcal{A}$ si riduce al problema $\mathcal{B}$ se

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

Quindi, se $A \leq_m B$:

1. Se B è ricorsivo, anche A è ricorsivo
2. Se A non è ricorsivo, anche B non è ricorsivo
3. Se B è RE, anche A è RE
4. Se A non è RE, anche B non è RE

La dimostrazione viene effettuata ragionando sulle funzioni caratteristiche/semi-caratteristiche dei due insiemi e combinandole con la funzione f di riduzione.

Ad esempio, si ha che A è RE $\Leftrightarrow A \leq_m K = \{x \,|x \in W_x\}$ (K è RE). questo perché

- ($\Leftarrow$): K è RE e quindi SC_K è calcolabile. Dal momento che f è funzione di riduzione, questa è calcolabile e $x \in A \Leftrightarrow f(x) \in K$, quindi si può definire $SC_A(x) = SC_K(f(x))$. La funzione semi-caratteristica di A è calcolabile perché è ottenuta componendo funzioni calcolabili e quindi anche A è RE.
- ($\Rightarrow$): A è RE e quindi $SC_A(x)$ è calcolabile. Bisogna trovare una funzione di riduzione $f : \mathbb{N} \to \mathbb{N}$ tale che $x \in A \Leftrightarrow f(x) \in K$.

  Definiamo

  $$g(x, y) = SC_A(x) = \begin{cases} 1 & x \in A \\ \uparrow & \text{altrimenti} \end{cases}$$

  dato che g è calcolabile, per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ tale che $\phi_{f(x)}(y) = g(x, y)$.

  Si ha quindi che f è proprio la funzione di riduzione $A \leq_m K$ perché:

  - Se $x \in A$: $\phi_{f(x)}(y) = 1$ e quindi $f(x) \in W_{f(x)}$, pertanto $f(x) \in K$.
  - Se $x \notin A$: $\phi_{f(x)}(y) = \uparrow$ e quindi $f(x) \notin W_{f(x)}$, pertanto $f(x) \notin K$.

8.5 Teorema di Rice

Ogni proprietà del comportamento di un programma non è decidibile.

Sia $A \subseteq \mathbb{N}$ una proprietà del comportamento di un programma (insieme saturo). Se la proprietà non è banale, l'insieme non è ricorsivo. Con proprietà banale si intende $A \neq \mathbb{N}$ e $A \neq \emptyset$.

8.5.1 Dimostrazione per riduzione

L'obiettivo è quello di dimostrare che $K = \{x \,|x \in W_x\} \leq_m A$. K è noto non essere ricorsivo.

Vogliamo quindi trovare una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $x \in K \Leftrightarrow f(x) \in A$.

Consideriamo un programma $e_0 \in \mathbb{N}$ tale che $\phi_{e_0} = \emptyset$ (ovvero che non termina mai). Questo programma può trovarsi in A o in $\overline{A}$.

- $e_0 \in \overline{A}$: Sia $e_1 \in A$ un programma qualsiasi, ed esiste perché $A \neq \emptyset$.

  Possiamo definire

  $$g(x,y) = \begin{cases} \phi_{e_1}(y) & x \in K \\ \phi_{e_0}(y) = \uparrow & x \notin K \end{cases} = \phi_{e_1}(y) \cdot \nVdash\big(\phi_x(x)\big)$$

  Questa funzione è calcolabile e quindi è possibile utilizzare il teorema SMN per trovare una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che

  $$g(x,y) = \phi_{f(x)}(y) \forall x, y$$

  e che può essere usata come funzione di riduzione.

  Questo perché:

  - Se $x \in K$ allora $\phi_{f(x)}(y) = \phi_{e_1}(y)\ \forall y$ e dato che A è saturo e che $e_1 \in A$, allora anche $f(x) \in A$ perché essendo A saturo contiene tutti gli indici che calcolano ϕ_{e_1}.
  - Se $x \notin K$ allora $\phi_{f(x)}(y) = \phi_{e_0}(y)\ \forall y$ e allo stesso modo dato che $e_0 \notin A$ anche $f(x) \notin A$, perché se $f(x) \in A$, allora anche e_0 dovrebbe appartenere ad A, ma questo è assurdo per ipotesi.

  Si ha quindi che f è la funzione di riduzione $K \leq_m A$ ed essendo K non ricorsivo, anche A **non è ricorsivo**.

- $e_0 \in A$: Sia $B = \overline{A}$, $B \neq \emptyset$ $B \neq \mathbb{N}$ e B è saturo perché è l'insieme complementare di un insieme saturo, ed infine $e_0 \notin B$. Per B valgono quindi tutte le condizioni del punto precedente, e quindi anche $B = \overline{A}$ non è ricorsivo.

8.6 Relazione tra R e RE

Un insieme $A \subseteq \mathbb{N}$ è ricorsivo se la sua funzione caratteristica $\mathcal{X}_A$ è calcolabile

$$\mathcal{X}_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

Invece è ricorsivamente enumerabile se la funzione semi-caratteristica SC_A è calcolabile:

$$SC_A(x) = \begin{cases} 1 & x \in A \\ \uparrow & x \notin A \end{cases}$$

Si ha quindi che $A \subseteq \mathbb{N}$ è ricorsivo $\Leftrightarrow$ A, $\overline{A}$ sono RE.

8.6.1 A ricorsivo $\Rightarrow$ A, $\overline{A}$ RE.

Se A è ricorsivo, la sua funzione caratteristica è calcolabile ed è definita come

$$\mathcal{X}_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

Si può quindi definire la funzione semi-caratteristica come minimalizzazione illimitata di $\mathcal{X}_A$:

$$SC_A(x) = \mathbb{1}\left(\mu w.\overline{sg}\Big(\mathcal{X}_A(x)\Big)\right)$$

così facendo si ottiene una funzione calcolabile che vale 1 se $x \in A$ (perché il segno negato vale 0) e risulta indefinita se $x \notin A$, perché la minimalizzazione non termina dato che il segno negato sarà sempre uguale a 1.

Per dimostrare che anche $\overline{A}$ è RE basta osservare che se A è ricorsivo, anche il suo complementare $\overline{A}$ è ricorsivo e quindi è possibile ripetere la stessa dimostrazione utilizzando $\mathcal{X}_{\overline{A}}$ per dimostrare che anche $\overline{A}$ è RE.

8.6.2 A, $\overline{A}$ RE $\Rightarrow$ A ricorsivo

Le due funzioni SC_A e $SC_{\overline{A}}$ sono calcolabili e quindi esistono due programmi e_1 ed e_2 le cui funzioni coincidono con le due funzioni semi-caratteristiche.

Si può quindi definire un programma che esegue i due programmi in parallelo e in base a quello che termina ritorna 0 oppure 1.

Questa funzione può essere definita come

$$\mathcal{X}_{\mathcal{A}}(x) = \left(\mu w.S\Big(e_1, x, (w)_1, (w)_2\Big) \vee S\Big(e_2, x, (w)_1, (w)_2\Big)\right)_1$$

ed è calcolabile perché definita per minimalizzazione illimitata.

Alcune osservazioni:

- La funzione $S(e, \vec{x}, y, t)$ vale 1 se il programma e su input x termina entro t passi producendo come output y.
- La minimalizzazione su w viene effettuata per simulare l'avanzamento dei passi. Vengono utilizzati gli esponenti della rappresentazione binaria di w perché così durante la minimalizzazione vengono provati tutti i possibili valori.
- Perché tutto funzioni è necessario che la funzione ϕ_{e_2} sia uguale a $\overline{\text{sg}}(SC_{\overline{A}}(x))$, perché deve valere 0 quando $x \in \overline{A}$.

8.7 Teorema di struttura

Sia $P(x)$ un predicato k-ario, $P(x)$ è semi-decidibile se e solo se esiste $Q(t, x)$ decidibile e tale che

$$P(\vec{x}) \equiv \exists t\, Q(t, \vec{x})$$

Ovvero il predicato P è una generalizzazione di un predicato decidibile che viene calcolato su più punti. In generale, quantificare esistenzialmente un predicato decidibile lo rende semi-decidibile.

Questo perché la funzione semi-caratteristica di P non farà altro che applicare la minimalizzazione illimitata alla funzione caratteristica di Q per cercare un valore che soddisfa Q:

$$SC_P(x) = \begin{cases} 1 \text{ se } P(x) \\ \uparrow \text{ altrimenti} \end{cases}$$
$$= \mathbb{1}\!\!\mathbb{K}\Big(\mu t.\text{“}Q(t,x)\text{”}\Big)$$
$$= \mathbb{1}\!\!\mathbb{K}\Big(\mu t.|\ \mathcal{X}_Q(t,x) - 1\ |\Big)$$

Viceversa, assumendo $P(x)$ semi-decidibile, si ha che SC_P è calcolabile, ovvero esiste un certo programma e tale che $\phi_e = SC_P$.

Quando il predicato $P(x)$ è vero, si ha che $SC_P(x) = 1 = \phi_e(x)$, ma questo vuol dire che il programma e termina su input x dopo un certo numero di passi t. Formalmente:

$$\phi_e(x) = 1 \Leftrightarrow \phi_e(x)\downarrow \Leftrightarrow \exists t\, H(e,x,t) = 1$$

Dove H è la funzione che ritorna 1 se la macchina e termina entro t passi sull'input x che sappiamo essere calcolabile.

Si ha quindi che, indicando il predicato $Q(t,x) \equiv \exists t\, H(e,x,t)$ è decidibile e $P(x) \equiv \exists t\, Q(t,\vec{x})$.

Utilizzando questo teorema è poi possibile andare a dimostrare il **teorema di proiezione**, ovvero che la classe RE è chiusa rispetto la quantificazione esistenziale. Lo si fa estraendo dal predicato semi-decidibile quello decidibile e poi si ragiona su come esprimere il predicato semi-decidibile come l'altro predicato semi-decidibile:

$$P'(\vec{y}) \equiv \exists x.\exists t.Q(t,x,\vec{y}) \equiv \exists w.\underbrace{Q((w)_1,(w)_2,\vec{y})}_{Q'}$$

8.8 Teorema di Rice-Shapiro

Le proprietà del comportamento (funzione calcolata) dei programmi possono essere decidibili se sono finitarie, ovvero dipendono da un numero finito di argomenti (es: *la funzione termina su 5*).

Più formalmente sia $\mathcal{A} \subseteq \mathbb{C}$ e $A = \{x \mid \phi_x \in \mathcal{A}\}$ (A saturo). Se A è RE, allora

$$\forall f\ \in \mathcal{C}.\Big(f \in A \Leftrightarrow \exists \vartheta \subseteq f \text{ e } \vartheta \in \mathcal{A}\Big)$$

A parole, sia $\mathcal{A}$ un sottoinsieme delle funzioni calcolabili e A l'insieme che contiene gli indici che calcolano queste funzioni. Se A è RE, allora una funzione f calcolabile appartiene ad $\mathcal{A}$ solo se esiste una sua parte finita che appartiene ad $\mathcal{A}$.

8.8.1 Dimostrazione

La dimostrazione si fa in due passi:

1. $\exists f, f \notin \mathcal{A}$ e $\exists \vartheta \subseteq f, \vartheta \in \mathcal{A} \Rightarrow A$ non è RE.
2. $\exists f, f \in \mathcal{A}$ e $\forall \vartheta \subseteq f, \vartheta \notin \mathcal{A} \Rightarrow A$ non è RE.

Caso 1 Sia $f \notin \mathcal{A}$ e $\vartheta \subseteq f, \vartheta \in \mathcal{A}$. Per provare che A non è RE viene fatta la riduzione $\overline{K} \leq_m A$.

Definiamo quindi

$$g(x,y) = \begin{cases} \vartheta(y) & x \in \overline{K} \\ f(y) & x \in K \end{cases} = \begin{cases} \uparrow & x \in \overline{K} \text{ e } y \notin dom(\vartheta) \\ f(y) = \vartheta(y) & x \in \overline{K} \text{ e } y \in dom(\vartheta) \\ f(y) & x \in K \end{cases}$$

$$= \begin{cases} f(y) & x \in K \text{ oppure } y \in dom(\vartheta) \\ \uparrow & \text{altrimenti} \end{cases}$$

Essendo $x \in K$ semi-decidibile, il predicato $Q(x,y) \equiv x \in K$ oppure $y \in dom(\vartheta)$ è semi-decidibile e pertanto la funzione g è calcolabile perché può essere definita come:

$$g(x,y) = f(y) \cdot SC_Q(x,y)$$

Essendo g calcolabile, per il teorema SMN esiste $S : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $g(x,y) = \phi_{S(x)}(y)$ che può funzionare da funzione di riduzione:

- $x \in \overline{K}$: $\phi_{S(x)}(y) = g(x,y)\ \forall y$ ed in particolare $= \vartheta(y)\ \forall y \Rightarrow \phi_{S(x)} = \vartheta$ ed essendo A saturo, $S(x) \in A$.
- $x \in K$: $\phi_{S(x)}(y) = g(x,y) = f(y)\ \forall y$ da ui segue che $\phi_{S(x)} = f \notin A$ e quindi dato che A è saturo, anche $S(x) \notin A$.

Caso 2 Sia $f \in \mathcal{A}$ e tutte le sue parti finite non sono in $\mathcal{A}$, voglio arrivare a dire che A non è RE. Anche in questo caso si fa la stessa riduzione.

$$g(x,y) = \begin{cases} f(y) & \text{se } \neg H(x,x,y) \\ \uparrow & \text{altrimenti} \end{cases}$$
$$= f(y) \cdot \not\vdash\big(\mu w.\mathcal{X}_H(x,x,y)\big)$$

Essendo g calcolabile, per SMN esiste S che funziona da funzione riduzione:

- $x \in \overline{K}$: $\forall y\ \neg H(x,x,y) = 0$ e quindi $\phi_{S(x)}(y) = f(y)\ \forall y$, ovvero $\phi_{S(x)} = f$ e dato che A è saturo e $f \in A$, anche $S(x) \in A$.
- $x \in K$: $\exists y_0$ tale che $H(x,x,y_0) = 1$, perché x termina in un certo numero di passi quando riceve se stesso in input. Tra tutti i possibili valori, prendo il minimo y_0 tale che $\forall y < y_0\ \neg H(x,x,y) = 1$ e $\forall y \geq y_0\ H(x,x,y) = 1$.

  Si ha quindi che $\phi_{S(x)}(y) = g(x,y) = \begin{cases} f(y) & y < y_0 \\ \uparrow & \text{altrimenti} \end{cases}$.

  Essendo che $\phi_{S(x)}$ è una parte finita di f, si ha che $\phi_{S(x)} \notin A$ e quindi $S(x) \notin A$.

8.9 Secondo teorema di Ricorsione

Sia $h : \mathbb{N} \to \mathbb{N}$ calcolabile e totale. Allora esiste $e \in \mathbb{N}$ tale che $\phi_e = \phi_{h(e)}$.

Capitolo 9

Soluzione compiti

9.1 Appello 2016-07-18

La soluzione non è del tutto corretta

9.1.1 Esercizio 1

Dimostrare il teorema di Rice.

Soluzione

Già fatto in altri appelli.

9.1.2 Esercizio 2

Data una funzione $f : \mathbb{N} \to \mathbb{N}$ si definisce il predicato $P(x, y) \equiv$ "$f(x) = y$", ovvero $P(x, y)$ è vero se e solo se $x \in dom(f)$ e $f(x) = y$. Dimostrare che la funzione f è calcolabile se e solo se $P(x, y)$ è semi-decidibile.

Soluzione

Se f è calcolabile c'è un programma e che la calcola.

$$SC_P(x, y) = \begin{cases} 1 & f(x) = y \\ \uparrow & \end{cases} = \nvDash\Big(\mu w.S(e, x, y, w)\Big)$$

SC_P è definita in modo calcolabile ed è corretta.
Se P è semi-decidibile, SC_P è calcolabile.

$$f(x) = \mu w.SC_P(x, w)$$

f è calcolabile. C'è qualche problema se la f non è iniettiva, ma se non ricordo male noi assumiamo che tutte le funzioni lo siano.

Possibile errore Sulla definizione di f può esserci un'errore. Se per un w non SC_P è $\uparrow$, f risulta indefinita su x anche quando non lo è.

Forse così va meglio (e è il programma che calcola SC_P):

$$f(x) = \Big(\mu w.S\Big(e, x, (w)_1, (w)_2\Big)\Big)_1$$

perché vengono fatti un po' di passi alla volta.

9.1.3 Esercizio 3

$$A = \{x | x \in W_x \wedge \phi_x(x) = x^2\}$$

Soluzione

A RE.

$$SC_A(x) = \not\vdash \left(\mu w. \left(S\Big(x, x, (w)_1, (w)_2\Big) \wedge (w)_1 = x^2 \right) \right)$$

A non ricorsivo, $K \leq_m A$.

$$g(x, y) = \begin{cases} y^2 & x \in K \\ \uparrow & \end{cases} = y^2 \cdot SC_K(x)$$

SMN.

- $x \in K$: $\phi_{f(x)} = y^2 \forall y.\ \phi_{f(x)}(f(x)) = f(x)^2 \to f(x) \in A$.
- $x \notin K$: $\phi_{f(x)}(y) = \uparrow \forall y.\ \phi_{f(x)}(f(x)) = \uparrow \to f(x) \notin A$.

A è RE, non ricorsivo, $\overline{A}$ è quindi non-RE.

Note Forse da qualche parte nel pdf c'è una soluzione migliore di questo esercizio.

9.1.4 Esercizio 4

$$B = \{x | \forall y \in W_x, \exists z \in W_x . (y < z) \wedge f(y) < f(z)\}$$

Soluzione

B è saturo, contiene tutte le funzioni strettamente crescenti.
$id, \emptyset \in \beta$, la funzione gradino e quella definita in un punto solo $\in \overline{\beta}$.

$$f(x) = \begin{cases} 0 & x = 0 \\ 1 & \text{altrimenti} \end{cases} = sg(x)$$

$$\vartheta(x) = \begin{cases} 0 & x = 0 \\ 1 & x = 1 \\ \uparrow & \end{cases}$$

$f \notin \beta$, $\vartheta \in \beta$, $\vartheta \subseteq f$, B è non-RE per Rice-Shapiro.

$$g(x) = \begin{cases} 0 & x = 0 \\ \uparrow & \text{altrimenti} \end{cases}$$

$id \notin \overline{\beta}$, $g \in \overline{\beta}$, $g \subseteq id$, $\overline{B}$ è non-RE per Rice Shapiro.

9.1.5 Esercizio 5

Dimostrare che non è saturo

$$C = \{x | x \in E_x\}$$

Soluzione

$$g(x,y) = \begin{cases} x & x = y \\ \uparrow & \end{cases} = x \cdot \not\Vdash\Big(\mu w.|x-y|\Big)$$

SMN + Secondo teorema

$$\phi_e(y) = \phi_{f(e)}(y) = g(e,y)$$

e

$$E_e = E_{f(e)} = \{e\}$$

Sia e' tale che $\phi_e = \phi_{e'}$, $E_{e'} = \{e\}$, $e' \notin C$, $e \in C$, l'insieme non è saturo.

9.2 Appello 2016-07-01

Correzione di Baldan.

9.2.1 Esercizio 1

Sia A un insieme ricorsivo e siano $f_1, f_2 : \mathbb{N} \to \mathbb{N}$ funzioni calcolabili. Dimostrare che è calcolabile la funzione $f : \mathbb{N} \to \mathbb{N}$ definita da

$$f(x) = \begin{cases} f_1(x) & x \in A \\ f_2(x) & x \notin A \end{cases}$$

Il risultato continua a valere se indeboliamo le ipotesi e assumiamo A RE? Spiegare come si adatta la dimostrazione, in caso positivo, o fornire un controesempio in caso negativo.

Soluzione

Siano e_1, e_2 i programmi che calcolano le due funzioni date.

La funzione f può essere calcolata con:

$$f(x) = \left(\mu w.\left(S\Big(e_1, x, (w)_1, (w)_2\Big) \wedge x \in A\right) \vee \left(S\Big(e_2, x, (w)_1, (w)_2\Big) \wedge x \notin A\right)\right)_1$$

questo perché l'appartenenza o meno ad A è decidibile in quanto A è ricorsivo.

Utilizzare

$$f(x) = f_1(x)\mathcal{X}_A(x) + f_2(x)\mathcal{X}_{\overline{A}}$$

è **sbagliato** perché una delle due funzioni può **non essere totale** e quindi verrebbe prodotto un risultato sbagliato, utilizzando una funzione sempre indefinita. Se ci fosse anche l'ipotesi della totalità il discorso sarebbe stato diverso.

Se invece A è ricorsivamente enumerabile, la non appartenenza ad A non è più decidibile.

Per dimostrare questo conviene trovare un contro esempio.

Posso prendere $A = K$, $f_1(x) = 1, f_2(x) = 0$, così facendo f risulta essere:

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

ed essendo f_1, f_2 calcolabili, anche $f = \mathcal{X}_K$ è calcolabile, ma è noto che questa funzione non è calcolabile e quindi in generale f non è calcolabile.

9.2.2 Esercizio 2

Dimostrare che un insieme A è RE se e solo se esiste una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile tale che $A = img(f) = \{f(x) : x \in \mathbb{N}\}$.

Soluzione

Se A è RE, allora la sua funzione SC_A è calcolabile ed è definita solo su A.

Per fare in modo che A sia l'immagine di f, si può prendere

$$f(x) = x \cdot SC_A(x)$$

In questo modo, quando $x \in A$, $SC_A(x) = 1$ e quindi $f(x) = x$ e pertanto $f(x) \in A$. f è calcolabile in quanto è ottenuta componendo funzioni calcolabili.

Se $A = img(f)$ con f calcolabile. L'idea è quella di definire la funzione semi-caratteristica di A in modo che valga 1 solo quando viene calcolata su un valore x che viene fornito in output dalla funzione.

Sia e un programma che calcola f. La funzione semi-caratteristica di A può essere definita come:

$$SC_A(x) = \not\vdash\Big(\mu w.S\Big(e, (w)_1, x, (w)_2, \Big)\Big)$$

È necessario effettuare la minimalizzazione perché f può non essere totale, ovvero **non** può essere definita come:

$$SC_A(x) = \not\vdash\Big(\mu w.|f(w) - x)|\Big)$$

9.2.3 Esercizio 3

$$A = \{x | x \in W_x \wedge \phi_x(x) > x\}$$

Soluzione

L'insieme è molto simile a K, quindi probabilmente è RE:

$$SC_A(x) = \not\vdash\Big(\mu w.x + 1 \dot{-} \phi_x(x)\Big) = \not\vdash\Big(\mu w.x + 1 \dot{-} \Phi_u(x, x)\Big)$$

La funzione è calcolabile, quindi A è RE e va bene anche se $x \notin W_x$ perché la funzione è correttamente indefinita.

Per dimostrare che A non è ricorsivo posso fare la riduzione $K \leq_m A$.

Serve quindi una funzione $g(x, y)$ tale che vista come funzione di y appartenga ad A solo se $x \in K$.

$$g(x, y) = \begin{cases} y + 1 & x \in K \\ \uparrow & x \notin K \end{cases} = (y + 1) \cdot SC_K(x)$$

Quando x in K, $g(x, x) = x + 1$ quindi la condizione di appartenenza ad A è soddisfatta.

Per SMN esiste f che funziona come funzione di riduzione e tale che $g(x, y) = \phi_{f(x)}(y)$:

- $x \in K$: $\phi_{f(x)}(y) = g(x, y) = y + 1 \forall y$ in particolare, $\phi_{f(x)}(f(x)) = f(x) + 1 \forall f(x)$ ed inoltre $f(x) \in W_{f(x)}$, quindi $f(x) \in A$.
- $x \notin K$: $\phi_{f(x)} = g(x, y) = \uparrow \forall y$, quindi $f(x) \notin W_{f(x)}$ e pertanto $f(x) \notin A$.

A non è ricorsivo ed è RE, quindi $\overline{A}$ è non RE.

9.2.4 Esercizio 4

$$B = \{x \in \mathbb{N} | \forall y \in W_x \exists z \in W_x \ (y < z) \wedge \phi_x(y) > \phi_x(z)\}$$

Soluzione

L'insieme contiene tutti gli indici, tali che per ogni valore del dominio, esiste un valore più grande tale che il valore della funzione calcolato in quel punto sia più basso.

L'insieme è saturo, perché

$$\beta = \{f \in \mathcal{C} : \forall y \in dom(f). \exists z \in dom(f). y < z \vee f(y) > f(z)\}$$

C'è una quantificazione universale, quindi è probabile che sia non RE.

Ad occhio c'è una sola funzione che appartiene a β è ed $\emptyset$ perché è sempre indefinita. Dato che $\emptyset$ è una parte finita di tutte le funzioni, basta trovarne una che non appartiene a β. Per non appartenere a β basta che una funzione non sia decrescente, quindi la funzione id non appartiene a β ma ammette parte finta che ci appartiene e quindi per Rice Shapiro, B è non RE.

$$\overline{\beta} = \{f \in \mathcal{C} : \exists y \in dom(f). \forall z \in dom(f). y > z \vee f(y) \leq f(z)\}$$

Osservando $\overline{\beta}$ a prima vista sembra essere non RE, però ragionando meglio sulle condizioni di appartenenza a $\overline{\beta}$ si può notare che viene richiesto che la funzione abbia un minimo e questo è sempre vero per le funzioni definite sui numeri naturali.

Quindi $f \in \overline{\beta} \Leftrightarrow f \neq \emptyset$, ovvero $\overline{\beta} = \mathcal{C} \setminus \{\emptyset\}$, $\beta = \{\emptyset\}$.

Si può definire la funzione semi-caratteristica:

$$SC_{\overline{B}}(x) = \not\Vdash\Big(\mu w.H\Big(x, (w)_1, (w)_2\Big)\Big)$$

Quindi $\overline{B}$ è RE e non ricorsivo, perché B è non-RE.

9.2.5 Esercizio 5

Enunciare il secondo teorema di ricorsione. Utilizzarlo per dimostrare che esiste un indice $e \in \mathbb{N}$ tale che $W_e = \{e^n \mid n \in \mathbb{N}\}$.

Soluzione

Il secondo teorema dice che, data una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, $\exists e \in \mathbb{N}$ tale che $\phi_e = \phi_{f(e)}$.

Definisco quindi una funzione che vista come funzione di y abbia le caratteristiche desiderate:

$$g(x, y) = \begin{cases} \log_x y & \text{se } y = x^n \text{ per qualche } n \\ \uparrow \end{cases} = \mu n.|x^n - y|$$

Essendo g calcolabile, per SMN esiste f calcolabile e totale, tale che $\phi_{f(x)}(y) = g(x, y)$.

Quindi $W_{f(x)} = \{x^n \mid n \in \mathbb{N}\}$. Per il secondo teorema di ricorsione $\exists e$ tale che $\phi_e = \phi_{f(e)}$.

9.3 Appello 2016-01-25

9.3.1 Esercizio 1

Dimostrare che $A \subseteq \mathbb{N}$ è ricorsivo se e solo se A e $\overline{A}$ sono RE.

Soluzione

Se A è ricorsivo posso definire

$$SC_A(x) = \nVdash\Big(\mu w.\overline{sg}\big(\mathcal{X}_A(x)\big)\Big)$$

che risulta calcolabile perché definita utilizzando funzioni calcolabili e quindi A è anche RE. Per ottenere $SC_{\overline{A}}$ basta togliere il segno negato, oppure utilizzare $\mathcal{X}_{\overline{A}}$.

Se invece sono entrambi RE, posso definire

$$\mathcal{X}_A(x) = \Big(\mu w.S\Big(e_1, x, \big(w\big)_1, \big(w\big)_2\Big) \vee S\Big(e_2, x, \big(w\big)_1, \big(w\big)_2\Big)\Big)_1$$

dove e_1 e e_2 sono i programmi che calcolano le due funzioni semi-caratteristiche (quella di $\overline{A}$ deve essere composta con il segno negato).

9.3.2 Esercizio 2

Definire una funzione $f : \mathbb{N} \to \mathbb{N}$ totale e non calcolabile tale che $f(x) = x$ per infiniti argomenti $x \in \mathbb{N}$ oppure dimostrare che questa funzione non esiste.

Soluzione

La funzione deve essere definita su tutto $\mathbb{N}$ e $f(x) = id(x) = x$.

Dato che le due funzioni sono uguali e sono definite sullo stesso dominio, un programma che calcola id calcola anche f ed essendo id calcolabile, esistono infiniti programmi che sono in grado di calcolarla e che quindi riescono a calcolare anche f.

Tuttavia, la funzione può essere non calcolabile:

$$f(x) = x \cdot \nVdash\Big(\mathcal{X}_K(x)\Big)$$

Così definita, è totale, uguale all'identità e non calcolabile perché $\mathcal{X}_K$ è non calcolabile.

9.3.3 Esercizio 3

$$A = \{x | \phi_x \text{ strettamente crescente}\}$$

Soluzione

A probabilmente è non-RE perché per valutare l'appartenenza è necessario controllare infiniti punti del dominio.

A è saturo:

$$\mathcal{A} = \{f | \forall x, y \in dom(f) : x < y \to f(x) < f(y)\}$$

Una qualsiasi funzione gradino:

$$f(x) = \begin{cases} 0 & x \leq x_0 \\ 1 & x > 0 \end{cases}$$

non appartiene a $\mathcal{A}$, ma la parte finita

$$\vartheta(x) = \begin{cases} 0 & x = x_0 \\ 1 & x = x_0 + 1 \\ \uparrow & \text{altriementi} \end{cases}$$

appartiene ad $\mathcal{A}$ e quindi per Rice-Shapiro, A è non-RE.

Per quanto riguarda $\overline{A}$, per decidere l'appartenenza basta trovare una coppia di punti sui quali la funzione è non crescente, quindi potrebbe essere RE.

$$SC_{\overline{A}}(x) = \not\Vdash\left(\mu w.\overline{sg}\Big(S\Big(x,(w)_1,(w)_2,(w)_3\Big)\wedge S\Big(x,(w)_4,(w)_5,(w)_6\Big)\wedge(w)_1<(w)_4\wedge(w)_2\geq(w)_5\Big)\right)$$

Essendo la funzione semi-caratteristica calcolabile, $\overline{A}$ è non-RE.

9.3.4 Esercizio 4

$$B = \{x : x > 0 \wedge x/2 \notin E_x\}$$

Soluzione

B probabilmente è non-RE, perché dovrei provare il programma x su infiniti input.

$$\overline{K} \leq_m B$$

$$g(x,y) = \begin{cases} \uparrow & x \notin K \\ y/2 & x \in K \end{cases} = y/2 \cdot SC_K(x)$$

Per SMN esiste la funzione di riduzione f:

- $x \notin K$: $\phi_{f(x)}(y) = g(x,y) = \uparrow \forall y \Rightarrow E_{f(x)} = \emptyset \Rightarrow f(x) \in B$.
- $x \in K$: $\phi_{f(x)}(y) = g(x,y) = y/2 \forall y \Rightarrow \phi_{f(x)}(f(x)) = f(x)/2 \Rightarrow f(x)/2 \in E_{f(x)} \Rightarrow f(x) \notin B$

E quindi B è non RE.
$\overline{B}$ invece sembra essere ricorsivo, perché basta eseguire in parallelo il programma su tutti i possibili input fino a che non viene trovato in output il valore $x/2$.

$$SC_{\overline{B}}(x) = \not\Vdash\left(\mu w.\overline{sg}\Big(S(x,(w)_1,x/2,(w)_2)\Big)\right)$$

$\overline{B}$ è RE e non ricorsivo.

9.3.5 Esercizio 5

$$\exists x.\phi_x(y) = x + y$$

Soluzione

Definisco $g(x,y) = x + y$.
Per SMN:

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

Per il secondo teorema di ricorsione

$$\exists x.\phi_x(y) = \phi_{f(x)}(y) = g(x,y) = x + y$$

9.4 Appello 2015-07-16

9.4.1 Esercizio 1 - Teorema di proiezione

$$P(x,\vec{y}) \text{ semi-dedibile} \Rightarrow \exists x\, P(x,\vec{y}) \text{ semi-decidibile}$$

Soluzione

Per il teorema di struttura: $P(\vec{x})$ è semi-decidibile $\Leftrightarrow$ esiste un predicato $Q(\vec{x}, y)$ tale che $P(\vec{x}) \equiv \exists y\, Q(\vec{x}, y)$.

Quindi essendo $P(x, \vec{y})$ è semi-decidibile, per il teorema di struttura esiste $Q(x, \vec{y}, z)$ decidibile.

Si può quindi riscrivere P' come:

$$P'(\vec{y}) \equiv \exists x\, \exists z Q(x, \vec{y}, z) \equiv \exists w.Q((w)_1, \vec{y}, (w)_2)$$

con Q decidibile e quindi per il teorema di struttura $P'(\vec{y})$ è semi-decidibile.

9.4.2 Esercizio 2 - Teoria sulla riduzione

$$A \text{ è RE } \Leftrightarrow A \leq_m K$$

Soluzione

($\Rightarrow$) Essendo K RE, la sua funzione semi-caratteristica $SC_K(x)$ è calcolabile ed esiste per ipotesi una funzione di riduzione $f : \mathbb{N} \to \mathbb{N}$.

Si possono quindi combinare queste due funzioni per definire la funzione semi-caratteristica di A:

$$SC_A(x) = SC_K(f(x))$$

la quale risulta calcolabile e quindi A è RE.

($\Leftarrow$) Per ipotesi A è RE, quindi la sua funzione semi-caratteristica è calcolabile.

Serve quindi una funzione di riduzione $f : \mathbb{N} \to \mathbb{N}$ per dimostrare che A si riduce a K.

Vogliamo quindi trovare una funzione tale che se $x \in A$, allora $f(x) \in K$, ovvero $\phi_{f(x)}(x) = \downarrow$ e che se $x \notin A$ allora $f(x) \notin K$ e quindi $\phi_{f(x)}(x) = \uparrow$.

Possiamo quindi definire la funzione

$$g(x, y) = \begin{cases} 1 & \text{se } x \in A \\ \uparrow & \text{altrimenti} \end{cases} = SC_A(x)$$

Questa funzione è calcolabile perché equivale alla funzione semi-caratteristica di A.

Possiamo quindi utilizzare il teorema SMN per trovare una funzione calcolabile e totale che ritorni un programma che calcoli g:

$$\phi_{f(x)}(y) = g(x, y)$$

La funzione f è quindi la funzione di riduzione cercata, perché:

- $x \in A$: $\phi_{f(x)}(y)$ è $\downarrow\ \forall y$ ed in particolare $\phi_{f(x)}(x) \downarrow$ e quindi il programma $f(x) \in K$
- $x \notin A$: $\phi_{f(x)}(y)$ è sempre indefinita ed in particolare $\phi_{f(x)}(x)$ è indefinita e quindi $f(x) \notin K$.

9.4.3 Esercizio 3 - Studio della ricorsività (riduzione)

$$A = \{x \in \mathbb{N} \mid \exists y \in E_x, \exists z \in W_x, x = y \cdot z\}$$

Soluzione

L'insieme A sembra essere RE perché per stabilire l'appartenenza ad A di una determinato programma è necessario andare ad eseguirlo su un certo numero di valori, fino a che non viene trovata una coppia che soddisfa la condizione.

La funzione semi-caratteristica può quindi essere scritta come:

$$SC_A(x) = 1\Big(\mu w.\Big(S\big(x,(w)_1,(w)_2,(w_3)\big) \wedge H\big(x,(w)_4,(w)_5\big) \wedge \big(x = (w)_2 \cdot (w)_4\big)\Big)\Big)$$

che risulta calcolabile e quindi A è RE.

Il primo termine cerca un numero di passi $(w)_3$ entro i quali il programma x produce un output $(w)_2 = y$ avendo in input $(w)_1$. In pratica prova vari valori di input per trovare i valori del dominio.

Il secondo termine cerca tutti i valori di input $(w)_4$ che fanno terminare il programma x in meno di $(w)_5$ passi.

A sembra inoltre essere non ricorsivo e questo può essere dimostrato riducendo $K \leq_m A$.

Si vuole quindi una funzione $f : \mathbb{N} \to \mathbb{N}$ tale che se $x \in K$, ovvero il programma x termina quando riceve in input se stesso, produca un programma $f(x)$ tale che ci siano almeno un elemento del dominio e uno del codominio che possono essere tra loro moltiplicati per ottenere l'indice del programma.

Viceversa se $x \notin K$, il programma $f(x)$ non deve avere questa coppia di valori nel dominio/codominio.

Osservazione: basta trovare un programma che termina producendo in output 1 quando in input riceve se stesso. In questo caso si ha che $1 \in E_x$ e che $x \in W_x$ e quindi la condizione di appartenenza ad A è soddisfatta.

Si può quindi definire la funzione

$$g(x,y) = \begin{cases} 1 & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases} = SC_K(x)$$

tale funzione è calcolabile e quindi per il teorema SMN esiste una funzione f tale che $\phi_{f(x)}(y) = g(x,y)$.

Questa funzione è proprio quella di riduzione perché:

- $x \in K$: $\phi_{f(x)}(f(x)) = 1$ e $f(x) \in W_{f(x)}$ e quindi $f(x) \in A$.
- $x \notin K$: $\phi_{f(x)}$ è sempre indefinita e quindi $E_{f(x)} = W_{f(x)} = \emptyset$, non è quindi possibile trovare y e z e pertanto $f(x) \notin A$.

Dal momento che A è RE e non è ricorsivo, $\overline{A}$ non è RE.

9.4.4 Esercizio 4 - Studio della ricorsività (Rice-Shapiro)

$$B = \{x \mid x \in \mathcal{B}\}$$
$$\mathcal{B} = \{f \mid |dom(f) \setminus cod(f)| \geq 2\}$$

Ovvero $\mathcal{B}$ contiene tutte le funzioni che hanno il dominio con più di due elementi diversi dal codominio.

Soluzione

Dal momento che $\mathcal{B}$ descrive una proprietà (non banale) di una funzione, l'insieme è saturo e quindi di sicuro B non è ricorsivo.

L'idea è quindi quella di applicare Rice-Shapiro per dimostrare che B non è RE. Per fare questo si può trovare una funzione $f \notin \mathcal{B}$ ma che ha almeno un parte finita $\vartheta \in \mathcal{B}$.

$$f(x) = x \dot{-} 2 = \begin{cases} 0 & x \leq 2 \\ x - 2 & \text{altrimenti} \end{cases}$$

In questo caso si ha $dom(f) = cod(f) = \mathbb{N}$, $|dom(f) \setminus cod(f)| = 0$ e quindi $f \notin \mathcal{B}$.

Come parte finita di f si può considerare

$$\vartheta(x) = \begin{cases} 0 & x \leq 2 \\ \uparrow & \text{altrimenti} \end{cases}$$

Si ha che $dom(\vartheta) = \{0, 1, 2\}$, $cod(f) = \{0\}$ e quindi $\vartheta \in \mathcal{B}$.

Abbiamo quindi una funzione $f \notin \mathcal{B}$ che ha una parte finita $\vartheta \subseteq f$ che appartiene a $\mathcal{B}$ e quindi per Rice-Shapiro B non è RE.

Resta da valutare $\overline{\mathcal{B}} = \{f \mid |dom(f) \setminus cod(f)| < 2\}$, che può essere fatto allo stesso modo.

Basta osservare che la funzione $1 \notin \overline{\mathcal{B}}$ e che ha come parte finita la funzione $\emptyset \in \overline{\mathcal{B}}$ e quindi sempre per Rice-Shapiro $\overline{\mathcal{B}}$ non è RE.

9.4.5 Esercizio 5 - Secondo teorema di ricorsione

Dimostrare che

$$\exists x.W_x = \{kx \mid k \in \mathbb{N}\}$$

Soluzione

Il secondo teorema di ricorsione afferma che data una funzione calcolabile e totale $h : \mathbb{N} \to \mathbb{N}$, esiste $e \in \mathbb{N}$ tale che $\phi_{h(e)} = \phi_e$.

Come prima cosa è necessario cercare una funzione che produca un programma $f(x)$ con le caratteristiche desiderate, ovvero che termini quando riceve un multiplo di se stesso in input.

Si può quindi definire la funzione

$$g(x, y) = \begin{cases} k & \text{se } y = kx \text{ per qualche } k \\ \uparrow & \text{altrimenti} \end{cases} = \mu k.|kx - y|$$

Essendo g calcolabile, per il teorema SMN, esiste una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che il programma $f(x)$ calcoli $\phi_{f(x)}(y) = g(x, y)$.

Si ha quindi che $W_{f(x)} = \{kx \mid k \in \mathbb{N}\}$, ma noi stiamo cercando un programma e.

Però la funzione f soddisfa le condizioni del secondo teorema di ricorsione, quindi $\exists e \in \mathbb{N}$ tale che $\phi_e = \phi_{f(e)}$ e quindi si ha che

$$W_e = W_{f(e)} = \{ke \mid k \in \mathbb{N}\}$$

9.5 Appello 2015-09-03

9.5.1 Esercizio 1 - Minimalizzazione illimitata

Definizione della minimalizzazione illimitata e dimostrazione della chiusura rispetto ad essa.

Soluzione

Vedi §**??**

9.5.2 Esercizio 2 - Funzione decrescente totale e non calcolabile

Esiste una funzione decrescente e non calcolabile?

Decrescente:

$$\forall x, y \in \mathbb{N}\, x \leq y \Rightarrow f(x) \geq f(y)$$

Soluzione

Tale funzione non esiste.

Sia f una funzione decrescente e totale. Si può quindi individuare $k = \min\{f(x) \mid x \in \mathbb{N}\}$ e $x_0 \in \mathbb{N}$ tale che $f(x_0) = k$.

Si ha quindi che $\forall x \geq x_0$, $f(x) \leq f(x_0) = k$ e quindi $f(x) = k$, perché k è il minimo valore assunto dalla funzione.

Si può quindi definire

$$\vartheta(x) = \begin{cases} f(x) & \text{se } x < x_0 \\ \uparrow & \text{altrimenti} \end{cases}$$

la quale essendo una parte finita di f è calcolabile. Con ϑ si può definire:

$$g(x) = \begin{cases} \vartheta(x) & \text{se } x < x_0 \\ k & \text{altrimenti} \end{cases}$$

che è calcolabile. Si potrebbe già concludere qui ma si può essere più precisi, sia e il programma che calcola ϑ:

$$g(x) = \mu w.\left(\Big(x < x_0 \wedge S\Big(e, x, (w)_1, (w)_2\Big)\Big) \vee \Big(x \geq x_0 \wedge (w)_1 = k\Big)\right)$$

9.5.3 Esercizio 3 - Ricorsività

Studiare la ricorsività di $A = \{x \mid E_x = W_x + 1\}$ sapendo che se $X \subseteq \mathbb{N}$, $X + 1 = \{x + 1 \mid x \in \mathbb{N}\}$

Soluzione

Si può osservare che l'insieme è saturo in quanto contiene tutte le funzioni il cui codominio è il successore del dominio.

Inoltre, probabilmente A non è RE perché per poter verificare l'appartenenza di una funzione all'insieme è necessario provare tutti i valori del dominio.

Si può quindi applicare Rice-Shapiro: la funzione id non appartiene a $\mathcal{A}$ perché il suo dominio coincide con il codominio e la funzione $\emptyset$ appartiene ad $\mathcal{A}$ perché sia il codominio che il dominio sono l'insieme vuoto.

Si ha quindi che $id \notin \mathcal{A}$, $\emptyset \in \mathcal{A}$ e $\emptyset$ è parte finita di id, quindi per il teorema di Rice-Shapiro A non è RE.

Resta da valutare $\overline{A}$ (saturo).

Posso definire la funzione

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

che ha codominio $\mathbb{N} - \{0\}$ e dominio $\mathbb{N}$, quindi $f \in \mathcal{A}$ e $f \notin \overline{\mathcal{A}}$.
Una parte finita di f è:

$$\vartheta(x) = \begin{cases} 1 & \text{se } x = 1 \\ \uparrow & \text{altrimenti} \end{cases}$$

e dato che $dom(\vartheta) = cod(\vartheta)$, $\vartheta \in \overline{\mathcal{A}}$ e quindi per Rice-Shapiro, $\overline{A}$ non è RE.

9.5.4 Esercizio 4 - Ricorsività

$$B = \{x \in \mathbb{N} \mid \forall y > x, 2y \in W_x\}$$

Soluzione

B non è saturo, in quanto descrive una proprietà non banale delle funzioni, inoltre, dato che per verificare l'appartenenza a B è necessario provare tutti i valori del dominio, probabilmente B non è RE.

Si tratta quindi di effettuare la riduzione $\overline{K} \leq_m B$, dove $\overline{K}$ è noto non essere RE e contiene tutti i programmi che non terminano su se stessi.

Serve quindi una funzione di riduzione che dato un programma x con $\phi_x(x) \uparrow$ fornisca un programma $f(x)$ tale che per tutti i valori maggiori del suo indice ($y > f(x)$), $2y \in W_{f(x)}$ e che se $x \in K$, $W_{f(x)}$ non contenga $2y$.

Possiamo quindi definire la funzione

$$g(x,y) = \begin{cases} 1 & x \in \overline{K} \\ \uparrow & \text{altrimenti} \end{cases} = 1\left(\mu w.|\mathcal{X}_{H(x,x,y)}|\right)$$

Trattandosi di una funzione calcolabile, per il teorema SMN esiste una funzione $f : \mathbb{N} \to \mathbb{N}$ tale che $\phi_{f(x)}(y) = g(x,y)$.

f è proprio la funzione di riduzione perché

- $x \in \overline{K}$: $\phi_{f(x)}(y) = 1\ \forall y$ e quindi $\phi_{f(x)}$ è definita su tutto $\mathbb{N}$, pertanto $f(x) \in B$.
- $x \in K$: $\phi_{f(x)}$ è sempre indefinita $\forall y$ e quindi anche se $y > f(x)$, $2y \notin W_{f(x)}$, pertanto $f(x) \notin B$.

Quindi B non è RE.

Per quanto riguarda $\overline{B}$, anche questo sembra non essere RE e si può provare la stessa riduzione $\overline{K} \leq_m \overline{B}$.

Serve quindi una funzione che dato un programma x che non termina quando riceve se stesso in input, fornisca un programma $f(x)$ tale che esiste un $y > x$, $2y \notin W_{f(x)}$ e che se x termina su se stesso in input, $f(x)$ termina su $2y$ per qualche $y > x$.

Possiamo quindi definire la funzione:

$$g(x,y) = \begin{cases} \uparrow & x \in \overline{K} \\ 1 & x \in K \equiv x \notin \overline{K} \end{cases} = SC_K(x)$$

g è calcolabile perché SC_K è calcolabile e quindi per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $\phi_{f(x)}(y) = g(x,y)$ ed è quindi la funzione di riduzione cercata, perché:

- $x \in \overline{K}$: $W_{f(x)} = \emptyset$ e quindi $f(x) \in \overline{B}$.
- $x \in K$: $W_{f(x)} = \mathbb{N}$ e quindi $\forall y > f(x)$, $2y \in W_{f(x)}$ e quindi $f(x) \notin \overline{B}$.

Segue quindi che anche $\overline{B}$ non è RE.

9.5.5 Esercizio 5 - Secondo teorema di ricorsione

Dimostrare che $f(x) = min\{y \mid \phi_x \neq \phi_y\}$ non è calcolabile.

Soluzione

Il secondo teorema di ricorsione dice che data $h : \mathbb{N} \to \mathbb{N}$ totale e calcolabile, $\exists e \in \mathbb{N}$ tale che $\phi_e = \phi_{h(e)}$.

La funzione f è totale perché fissato un programma x è sempre possibile trovare un programma y che calcola una funzione diversa.

Inoltre, per come è definita f, questa non ha punti fissi, perché fissato un e, $\phi_{f(e)} \neq \phi_e$.

Quindi per il secondo teorema di ricorsione, f non può essere calcolabile perché altrimenti dovrebbe esistere un punto fisso.

9.6 Appello 2015-06-30

Fatta da Baldan a lezione

9.6.1 Esercizio 1

Dimostrazione del teorema di struttura.

9.6.2 Esercizio 2

$A, B \subseteq \mathbb{N}$, $A \leq_m B$ se esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che $x \in A$ se e solo se $f(x) \in B$.

È vero che per ogni $A \subseteq \mathbb{N}$, $A \leq_m A \cup \{0\}$? Che condizioni possono essere aggiunte perché questo sia vero?

Soluzione

Sembra ragionevole pensare che se si è in grado di decidere l'appartenenza a $A \cup \{0\}$, sia possibile definire anche l'appartenenza per A.

Conviene quindi provare questa strada, ovvero cercare una funzione f tale che se $x \in A$, allora $f(x) \in A \cup \{0\}$.

Se $x = 0$ e $x \in \overline{A}$, devo mandarlo in $\overline{A \cup \{0\}}$.

$$f(x) = \begin{cases} x & x \neq 0 \\ x_0 & x = 0 \end{cases}$$

Riassumendo, c'è un caso banale se $0 \in A$, perché $A \cup \{0\} = A$ e quindi $A \leq_m A$ e come funzione di riduzione posso utilizzare l'identità.

Se invece $0 \in \overline{A}$, basta prendere un elemento $x_0 \neq 0 \notin A$ e ritornare quello, ovvero la funzione di riduzione risulta essere la f sopra riportata.

Resta da dimostrare che questa funzione è calcolabile e che funzioni da funzione di riduzione. Ovviamente è calcolabile perché può essere definita come

$$f(x) = x \cdot \mathrm{sg}(x) + x_0 \cdot \overline{\mathrm{sg}}(x)$$

Per dimostrare che f è funzione di riduzione da $A \leq_m A \cup \{0\}$:

1. $x \in A$ ovvero $x \neq 0$, $f(x) = x \in A \cup \{0\}$ ed è corretto.
2. $x \notin A$, devo distinguere i due casi, se $x \neq 0$ si ha $f(x) = x$, ovvero $x \in \overline{A \cup \{0\}}$ ed è quello che vogliamo. Se invece $x = 0$, $f(x) = x_0$ e per costruzione $x_0 \in \overline{A \cup \{0\}}$.

f si comporta quindi come una funzione di riduzione, ma funziona perché esiste x_0 e quindi il ragionamento fatto finora vale $A \neq \mathbb{N} - \{0\}$.

Se $A = \mathbb{N} - \{0\}$, si ha che $A \cup \{0\} = \mathbb{N}$ e quindi non è possibile trovare un'immagine in $\overline{A \cup \{0\}}$ per 0.

Tornando alla domanda, non è sempre vero che $A \leq_m A \cup \{0\}$, perché se $A = \mathbb{N} - \{0\}$ una funzione di riduzione dovrebbe assicurare che $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale che $x \in A$ se e solo se $f(x) \in A \cup \{0\}$ perché non si riesce a trovare $f(0) \notin \mathbb{N}$.

Per rispondere alla seconda parte si ha che $A \neq \mathbb{N} - \{0\}$ è condizione necessaria e sufficiente per far valere la riduzione: **sufficiente** perché $A \neq \mathbb{N} - \{0\} \Rightarrow A \leq_m A \cup \{0\}$, due casi:

- Se $0 \in A$: la riduzione può essere fatta trivialmente dalla funzione identità.
- Se $0 \notin A$: sia $x_0 \neq 0$, $x_0 \notin A$ ed x_0 esiste per ipotesi. Così facendo è possibile definire $f(x)$ come prima e per quanto dimostrato è funzione di riduzione.

La condizione è anche **necessaria** perché sennò vale il controesempio iniziale.

In alternativa, un'altra condizione solamente sufficiente è quella che $0 \in A$, è più stretta rispetto la prima e quindi è una risposta peggiore.

9.6.3 Esercizio 3

Gli esercizi 3 e 4 sono spesso di questo tipo.

Studiare la ricorsività dell'insieme $A = \{x \in \mathbb{N} \mid x \in E_x \cup W_x\}$, ovvero dire se A è ricorsivo o ricorsivamente enumerabile e com'è il suo complementare.

Soluzione

Per sapere se $x \in E_x$ posso provare tutti gli input, e questo si può fare anche se in alcuni input non termina, andando ad eseguire un po' di passi per ognuno dei valori fino a che non termina. Allo stesso modo posso provare se $x \in W_x$.

Sembrerebbe quindi che A sia ricorsivamente enumerabile.

Per dimostrare che A non è ricorsivo posso provare a ridurre $K \leq_m A$.

Se avessi un programma P che prende in input x e risponde alla domanda $x \in A$? SI/NO, potrei usare questo programma per risolvere l'halting problem, perché potrei estendere il programma in modo che prima venga eseguito $P_x(x)$ su un input y non rilevante e poi fornire il risultato a P.

Voglio quindi una funzione f calcolabile e totale, tale che $x \in K$ se e solo se $f(x) \in A$.

Definisco quindi

$$g(x,y) = \begin{cases} \phi_x(x) & x \in K \\ \uparrow & \text{altrimenti} \end{cases} = \Phi_U(x,x)$$

questa funzione è calcolabile e quindi per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $\phi_{f(x)}(y) = g(x,y)$.

f è funzione di riduzione $K \leq_m A$ perché se:

- $x \in K$: $\phi_{f(x)}(y) = g(x,y) = \phi_x(x) \downarrow \forall y$. In particolare $W_{f(x)} = \mathbb{N}$ e quindi per ogni valore di x lo trovo nel dominio della funzione e quindi sta anche nell'unione tra il dominio e il codominio e quindi $f(x) \in A$.
- $x \notin K$: $\phi_{f(x)}(y) = g(x,y) \uparrow \forall y$, ovvero $W_{f(x)} = \emptyset$ e $E_{f(x)} = \emptyset$ e quindi di sicuro $f(x)$ non compare nell'unione dei due, in particolare $f(x) \notin A$.

Dato che K si riduce ad A e che K non è ricorsivo, anche A non è ricorsivo.

Per dimostrare che A è RE si può provare a ridurre $A \leq_m K$ ma è sconsigliato, oppure si può definire la funzione semi-caratteristica di A come funzione calcolabile. Ovvero:

$$SC_A(x) = \begin{cases} 1 & x \in A \\ \uparrow & \text{altrimenti} \end{cases} = p(x) = 1\Big(\mu w.\underbrace{S(x,(w)_1,x,(w)_2)}_{x\in E_x} \vee \underbrace{H(x,x,(w)_3)}_{x\in W_x}\Big)$$

$p(x)$ è quindi la funzione semi caratteristica perché[1]:

- $x \in A$: $x \in E_x$ oppure $x \in W_x$
  - $x \in E_x$: esiste z tale che $\phi_x(z) = x$ e quindi esiste t tale che $S(x,z,x,t)$ e quindi esiste anche un w le cui componenti prime producono lo stesso risultato
  - $x \in W_x$: esiste t tale che $H(x,x,t)$, quindi esiste w tale che $(w)_3 = t$.

  In entrambi i casi la minimalizzazione termina e produce $p(x) = 1$. Da notare che le osservazioni effettuate sono invertibili e quindi valgono tutte in se e solo se.

La funzione $p(x)$ è ovviamente calcolabile per minimalizzazione illimitata.

Possiamo quindi dire che A non è ricorsivo e che è RE, quindi per forza di cose $\overline{A}$ non è RE, perché altrimenti A sarebbe ricorsivo.

9.6.4 Esercizio 4

Studiare la ricorsività dell'insieme $B = \{x \in \mathbb{N} \mid 1 \leq |E_x| \leq 2\}$. Dire se è saturo.

Soluzione

L'insieme B può essere definito come un insieme di programmi che calcolano funzioni con un codominio di cardinalità 1 o 2, quindi per definizione è saturo.

$$B = \{x \in \mathbb{N} | \phi_x \in \mathcal{B}\}$$
$$\mathcal{B} = \{f \in \mathcal{C} | 1 \leq |cod(f)| \leq 2\} \subseteq \mathcal{C}$$

B probabilmente non è RE perché per determinare se un programma appartiene a B è necessario provare tutti i possibili valori di input e verificare che in output vengono ottenuti al massimo 2 valori distinti.

Considero la funzione

$$f(x) = x \dot{-} 2$$

Si ha che $cod(f) = \mathbb{N}$ e quindi $f \notin \mathcal{B}$.

Possiamo trovare una funzione

$$\vartheta(x) = \begin{cases} x \dot{-} 2 & x \leq 2 \\ \uparrow & \text{altrimenti} \end{cases}$$

Per come è definita $\vartheta \subseteq f$, è finita e $cod(\vartheta) = \{0\}$, ovvero ha cardinalità 1 e quindi $\vartheta \in \mathcal{B}$. Abbiamo quindi una funzione f che non sta in $\mathcal{B}$ ma che ammette una parte finita in $\mathcal{B}$ e quindi per il teorema di Rice Shapiro, B non è RE.

Per quanto riguarda $\overline{B}$, $\overline{\mathcal{B}} = \{f \mid |cod(f)| < 1 \text{ oppure } |cod(f)| > 2\}$, la funzione $g = \vartheta \notin \overline{\mathcal{B}}$ e $\vartheta \subseteq g$, ϑ finita e $\vartheta \notin \overline{\mathcal{B}}$ si ha che $\overline{B}$ non è RE.

Quindi sia B che $\overline{B}$ non sono RE e quindi non sono neanche ricorsivi[2].

[1] Questa è opzionale, diciamo che se c'è del tempo da perdere conviene farla

[2] Specificarlo è bene

9.6.5 Esercizio 5

Tipicamente è un'applicazione del secondo teorema di ricorsione.

Dato l'insieme $C = \{x \in \mathbb{N} \,|\, [0, x] \subseteq W_x\}$, enunciare il secondo teorema di ricorsione ed utilizzarlo per dimostrare che l'insieme C non è saturo.

Soluzione

Siccome dipende da una proprietà di un programma, è probabile che non sia saturo.

L'idea è quella di trovare un certo indice e tale che $W_e = [0, e]$. Se riusciamo a trovare un programma del genere abbiamo sostanzialmente concluso, perché siamo certi che esistono infiniti indici che calcolano la stessa funzione ϕ_e e tra tutti questi ce ne è sicuramente uno che è più grande di e, ovvero esiste $e' \in \mathbb{N}$ tale che $\phi_{e'} = \phi_e$ e $e' > e$.

Trovato questo e' abbiamo $[0, e'] \nsubseteq W_{e'} = W_e = [0, e]$ e quindi $e' \notin C$ e pertanto C non è saturo.

Dimostriamo che e esiste e per farlo consideriamo la funzione

$$g(x, y) = \begin{cases} 0 & y \in [0, x] \\ \uparrow & \text{altrimenti} \end{cases} = \mu z.y \dot{-} x$$

questa funzione è calcolabile e pertanto per il teorema SMN esiste $S : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che $g(x, y) = \phi_{S(x)}(y) \forall x, y$.

Per il secondo teorema della ricorsione esiste un indice e tale che $\phi_e = \phi_{S(e)}$ e quindi

$$\phi_e(y) = \phi_{S(e)}(y) = g(e, y) = \begin{cases} 0 & y \in [0, x] \\ \uparrow & \text{altrimenti} \end{cases}$$

ovvero $W_e = [0, e]$ e quindi e è proprio l'indice che cercavamo.

9.7 Appello 2014-08-25

9.7.1 Esercizio 1

Enunciare e dimostrare il teorema di Rice.

Soluzione

Ogni proprietà non banale relativa al comportamento dei programmi non è decidibile.

Ovvero sia $A \subseteq \mathbb{N}$ l'insieme dei programmi che hanno una qualche proprietà e quindi A è un insieme saturo, se $A \neq \mathbb{N}$ e $\neq \emptyset$, allora A non è ricorsivo.

Questo può essere dimostrato per riduzione $K \leq_m A$.

Sia e_0 un programma che calcola la funzione sempre indefinita ($\phi_{e_0} = \emptyset$). Questo programma può essere in A o in $\overline{A}$. Assumiamo che sia in $\overline{A}$.

Possiamo quindi scegliere un qualsiasi programma $e_1 \in A$ e almeno uno deve esserci perché $A \neq \emptyset$.

Questi due programmi possono essere utilizzati per definire la funzione

$$g(x, y) = \begin{cases} \phi_{e_1}(y) & \text{se } x \in K \\ \phi_{e_0}(y) & \text{se } x \notin K \end{cases} = \begin{cases} \phi_{e_1}(y) & \text{se } x \in K \\ \uparrow & \text{se } x \notin K \end{cases} = \phi_{e_1}(y) \cdot \nVdash\big(\phi_x(x)\big)$$

la quale è calcolabile e quindi per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ tale che $g(x, y) = \phi_{f(x)}(y)$ e che può essere usata come funzione di riduzione, perché:

- Se $x \in K$: $\phi_{f(x)} = \phi_{e_1}$ e quindi dal momento che A è saturo e che $e_1 \in A$, allora anche $f(x) \in A$.

- Se $x \notin K$: $\phi_{f(x)} = \phi_{e_0}$ e quindi dal momento che $e_0 \notin A$, anche $f(x) \notin A$, perché se $f(x)$ fosse in A, anche e_0 dovrebbe esserci, perché A è saturo e i due programmi calcolano la stessa funzione.

Se invece $e_0 \in A$, basta osservare che il complementare di un insieme saturo è saturo e, indicando con $B = \overline{A}$, $\overline{B} = A$, si ha che $e_0 \in \overline{B}$ e quindi vale la dimostrazione precedente.

9.7.2 Esercizio 2

Può esistere una funzione **non calcolabile** $f : \mathbb{N} \to \mathbb{N}$ tale che **per ogni** funzione non calcolabile $g : \mathbb{N} \to \mathbb{N}$, la funzione h definita come $h(x) = f(x) + g(x)$ sia calcolabile? Motivare la risposta, fornendo un'esempio di f oppure dimostrando che non può esistere.

Soluzione

Supponiamo che f esista e che quindi $h(x) = f(x) + g(x)$ sia calcolabile. Dal momento che h è calcolabile per tutte le g non calcolabili, si ha anche che $h(x) = f(x) + f(x) = 2 \cdot f(x)$ deve essere calcolabile.

Si ha quindi che f può essere definita come

$$f(x) = qt\big(h(x), 2\big)$$

f risulta quindi essere calcolabile per composizione di funzioni calcolabili il che va contro l'ipotesi della non calcolabilità di f e pertanto f non può esistere.

9.7.3 Esercizio 3

Studiare la ricorsività dell'insieme $A = \{x \mid \phi_x(y + x) \downarrow \text{ per qualche } y \geq 0\}$

Soluzione

A sembra essere RE, perché per determinare l'appartenenza è necessario determinare se il programma x termina ricevendo in input se stesso più una qualche costante.

$$SC_A(x) = \begin{cases} 1 & x \in A \; \exists y \geq 0 \mid \phi_x(x + y) \downarrow \\ \uparrow & x \notin A \end{cases} = \not\vdash\Big(\mu w.\overline{sg}\Big(S\big(x, x + (w)_1, (w)_2, (w)_3\big)\Big)\Big)$$

La funzione semi-caratteristica è calcolabile per composizione di funzioni calcolabili e quindi A è RE.

Per provare che A non è ricorsivo, si può effettuare la riduzione $K \leq_m A$.

Ovvero bisogna trovare una funzione $f : \mathbb{N} \to \mathbb{N}$ che dato $x \in K$, $f(x) \in A$ e se $x \notin K$, $f(x) \notin A$.

Si può osservare che se $x \in K$, $\phi_x(x) \downarrow$ e quindi anche $\phi_x(x + 0) \downarrow$ e pertanto x è anche in A.

Si può quindi definire la funzione

$$g(x, y) = \begin{cases} \phi_x(x) & x \in K \\ \uparrow & x \notin K \end{cases} = \phi_x(x) = \Phi_U(x, x)$$

la quale è calcolabile e quindi per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile, totale e tale che $g(x, y) = \phi_{f(x)}(y)$.

f è funzione di riduzione perché:

- $x \in K$: $\phi_{f(x)}(z) = g(x, z) = \phi_x(x) = \downarrow \; \forall z$ e quindi anche $\phi_{f(x)}(f(x) + 0) \downarrow$ e quindi $f(x) \in A$ per $y = 0$.

- $x \notin K$: $\phi_{f(x)}(z) = g(x,z) = \uparrow \ \forall z$ e quindi non esiste $y \geq 0$ tale che $\phi_{f(x)}(f(x)+y)\downarrow$ e quindi $f(x) \notin A$.

In alternativa potevo usare

$$g(x,y) = \begin{cases} 1 & x \in K \\ \uparrow & x \notin K \end{cases} = SC_K(x)$$

Essendo A non ricorsivo e RE, $\overline{A}$ non è RE.

9.7.4 Esercizio 4

Studiare la ricorsività dell'insieme $B = \{x \mid Pr \subseteq W_x\}$, dove $Pr \subseteq \mathbb{N}$ è l'insieme dei numeri primi.

Soluzione

Si può osservare che l'insieme B è saturo perché riguarda una proprietà della funzione calcolata dai programmi:

$$B = \{x \mid \phi_x \in \beta\}$$
$$\beta = \{f \mid Pr \subseteq dom(f)\}$$

Inoltre si può osservare che B probabilmente non è RE perché per determinare l'appartenenza bisogna verificare infiniti valori del dominio.

La funzione id appartiene a β in quanto è definita su tutto $\mathbb{N}$ e quindi anche su tutti i numeri primi, mentre tutte le sue parte finite non appartengono a β perché hanno un dominio finito, il quale non può contenere tutti gli infiniti numeri primi.

Si ha quindi una funzione che appartiene all'insieme e tutte le sue parti finite che non ci appartengono, quindi per Rice-Shapiro B non è RE.

Per quanto riguarda $\overline{B}$, si ha che contiene tutti i programmi che calcolano funzioni definite su qualche numero primo, anche nessuno (ma non tutti) ed è sempre un insieme saturo.

Analogamente a prima, la funzione id non appartiene a $\overline{\beta}$ e la funzione

$$\vartheta(x) = \begin{cases} x & \text{se } x \leq 5 \\ \uparrow & \text{altrimenti} \end{cases}$$

è una parte finita di id e appartiene a $\overline{\beta}$. Si ha quindi che per il teorema di Rice-Shapiro anche $\overline{B}$ non è RE.

9.7.5 Esercizio 5

Dimostrare che esiste $n \in \mathbb{N}$ tale che $\phi_n = \phi_{n+1}$ ed esiste anche $m \in \mathbb{N}$ tale che $\phi_m \neq \phi_{m+1}$.

Soluzione

Il programma composto dalla sola istruzione `Z(1)` viene codificato con $2^{4\cdot(1-1)+1} - 2 = 0$ mentre il programma `Z(1)Z(1)` viene codificato con $2^0 \cdot 3^1 - 2 = 1$, entrambi i programmi calcolano la funzione $f(x) = 0$ e quindi il primo caso è dimostrato.

Il programma composto dall'istruzione `S(1)` viene codificato con $2^{(4(1-1)+1)+1} - 2 = 2$ e calcola la funzione $f(x) = x + 1$ che è diversa dalla funzione calcolata dal programma di indice 1.

Ricapitolando:

- $\phi_0(x) = 0$
- $\phi_1(x) = 1$

- $\phi_2(x) = x + 1$

Segue quindi che esistono $n = 0$ e $m = 1$ che soddisfano quanto richiesto.

Soluzione decisamente migliore

La funzione *succ* è una funzione calcolabile e totale, quindi per il secondo teorema di ricorsione $\exists e \in \mathbb{N}$ tale che $\phi_e = \phi_{succ(e)} = \phi_{e+1}$.

Tuttavia deve esistere almeno un e tale che $\phi_e \neq \phi_{e+1}$ perché se così non fosse, tutti i programmi calcolerebbero la stessa funzione e quindi solamente una funzione sarebbe calcolabile, ma è noto che ci sono più di due funzioni calcolabili.

9.8 Appello 2014-07-21

9.8.1 Esercizio 1

Enunciare e dimostrare il teorema di Rice

Soluzione

Il teorema di Rice asserisce che le proprietà non banali di un programma non sono ricorsive.

Ovvero dato $A \subseteq \mathbb{N}$ saturo e tale che $A \neq \mathbb{N}$ e $\neq \emptyset$, A è ricorsivo.

Questo si dimostra per riduzione $K \leq_m A$.

Sia e_0 un programma che calcola la funzione sempre indefinita ($\phi_{e_0} = \emptyset$). Questo programma può essere in A o in $\overline{A}$. Assumiamo che sia in $\overline{A}$.

Sia e_1 un programma in A, ed esiste perché $A \neq \emptyset$.

Serve quindi una funzione tale che $x \in K \Leftrightarrow f(x) \in A$.

Possiamo definire

$$g(x,y) = \begin{cases} \phi_{e_1}(y) & x \in K \\ \phi_{e_0}(y) = \uparrow \forall y & x \notin K \end{cases} = \phi_{e_1} \cdot \not\Vdash\big(SC_K(x)\big)$$

g è calcolabile e totale, quindi per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, tale che $g(x,y) = \phi_{f(x)}(y)$.

f è funzione di riduzione perché

- $x \in K$: $\phi_{f(x)}(y) = g(x,y) = \phi_{e_1}(y)\ \forall y$ e quindi dal momento che A è saturo e che $e_1 \in A$, anche $f(x) \in A$.
- $x \notin K$: $\phi_{f(x)}(y) = g(x,y) = \uparrow = \phi_{e_0}(y)\ \forall y$ e quindi, dato che $e_0 \notin A$, anche $f(x) \notin A$, perché se $f(x) \in A$, anche e_0 dovrebbe essere in A dato che calcolano la stessa funzione.

Segue quindi che A non è ricorsivo, se $e_0 \notin A$. Assumendo invece che $e_0 \in A$, si può osservare che il complementare di un insieme saturo è anch'esso saturo e che se $B = \overline{A}$, $\overline{B} = A$ e $e_0 \in \overline{B}$ e quindi vale la dimostrazione precedente.

9.8.2 Esercizio 2

Esiste una funzione totale non calcolabile $f : \mathbb{N} \to \mathbb{N}$ tale che la funzione $g : \mathbb{N} \to \mathbb{N}$ definita per ogni $x \in \mathbb{N}$, da $g(x) = f(x) \dot{-} x$ sia calcolabile? Fornire un esempio di f oppure dimostrare che tale funzione non esiste.

Soluzione

Si esiste:

$$f(x) = \mathcal{X}_K(x) = \begin{cases} 1 & x \in K \\ 0 & x \notin K \end{cases}$$

(funzione caratteristica dell'insieme K, è noto che non è calcolabile).

Con questa funzione si ha che:

$$g(x) = \begin{cases} 0 & x \geq 1 \\ 1 & x = 0 \text{ e } f(x) = 1 \\ 0 & x = 0 \text{ e } f(x) = 0 \end{cases} = \begin{cases} 0 & x \geq 1 \\ 1 & x = 0 \text{ e } x \in K \\ 0 & x = 0 \text{ e } x \notin K \end{cases} = \begin{cases} 0 & x \geq 1 \\ 1 & x = 0 \text{ e } \phi_x(x) \downarrow \\ 0 & x = 0 \text{ e } \phi_x(x) \uparrow \end{cases}$$

g risulta quindi essere calcolabile in quanto viene definita per casi e utilizzando solamente funzioni calcolabili. $\phi_x(x)$ può essere calcolato utilizzando la funzione universale.

verificare il caso in cui f(x) vale 0

9.8.3 Esercizio 3

Una funzione parziale è iniettiva quando per ogni $x, y \in dom(f)$, se $f(x) = f(y)$, allora $x = y$. Studiare la ricorsività di $A = \{x | \phi_x \text{è iniettiva}\}$.

Soluzione

L'insieme è saturo perché descrive una proprietà non banale delle funzioni calcolate dai programmi che appartengono all'insieme

Probabilmente A non è RE, perché per provare se una funzione è iniettiva è necessario verificare tutti i valori del dominio.

La funzione

$$f(x) = \begin{cases} x & x \leq 3 \\ 0 & \text{altrimenti} \end{cases}$$

non è iniettiva e quindi non appartiene ad $\mathcal{A}$, ma ammette una parte finita $\vartheta \in \mathcal{A}$:

$$\vartheta = \begin{cases} x & x \leq 3 \\ \uparrow & \text{altrimenti} \end{cases}$$

e quindi per il teorema di Rice Shapiro, A non è RE.

Per quanto riguarda $\overline{A}$ non si può fare lo stesso ragionamento perché una funzione iniettiva ha tutte le parti finite iniettive, e sembra valere anche il viceversa.

Per vedere se una funzione non è iniettiva, basta trovare x, y tali che $x \neq y$ e $f(x) = f(y)$ e quindi stabilire l'appartenenza ad $\overline{A}$ sembra essere semi-decidibile.

$$SC_{\overline{A}}(x) = \nVdash \left(\mu w. \left(\underbrace{S\Big(x, (w)_1, (w)_2, (w)_3\Big)}_{a=(w)_1 \in W_x} \wedge \underbrace{S\Big(x, (w)_4, (w)_5, (w)_6\Big)}_{b=(w_4) \in W_x} \wedge \Big(\underbrace{(w)_1 \neq (w)_4}_{a \neq b} \Big) \wedge \Big(\underbrace{(w)_2 = (w)_5}_{\phi_x(a) = \phi_x(b)} \Big) \right) \right)$$

$SC_{\overline{A}}$ è calcolabile per composizione di funzioni calcolabili (quasi, l'uguaglianza e la disuguaglianza sono predicati, ma la loro funzione caratteristica è calcolabile) e quindi $\overline{A}$ è RE.

9.8.4 Esercizio 4

Studiare la ricorsività di $B = \{x | x \in W_x \setminus \{0\}\}$

Soluzione

B sembra essere RE, perché molto simile a K.

$$SC_B(x) = \begin{cases} 1 & x \in K \wedge x \neq 0 \\ \uparrow & x = 0 \\ \uparrow & x \notin K \end{cases} = \not\Vdash\big(\mu w.\overline{sg}(x)\big) \cdot SC_K(x)$$

SC_B è calcolabile per composizione di funzioni calcolabili e quindi B è RE.

$\overline{B} = \{x | \phi_x(x) \uparrow\} \cup \{0\} = \overline{K} \cup \{0\}$ e quindi $\overline{K} \cup \{0\} \subseteq \overline{B}$. Essendo $\overline{K}$ non RE si può effettuare la riduzione $\overline{K} \leq_m \overline{B}$ utilizzando come funzione la funzione identità e quindi anche $\overline{B}$ non è RE.

9.8.5 Esercizio 5

Enunciare il secondo teorema di ricorsione e dimostrare che esiste $n \in \mathbb{N}$ tale che $W_n = E_n = \{kn \mid k \in \mathbb{N}\}$

Soluzione

Il teorema asserisce che data una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, $\exists e \in \mathbb{N}$ tale che $\phi_e = \phi_{f(e)}$.

Definisco

$$g(x,y) = \begin{cases} x \cdot y & \text{se } x \text{ è multiplo di } y \\ \uparrow & \text{altrimenti} \end{cases} = xy \cdot \not\Vdash\big(\mu z.|zx - y|\big)$$

ed è calcolabile perché definita utilizzando funzioni calcolabili, quindi per SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che $g(x,y) = \phi_{f(x)}(y) \forall y$ e tale che $W_{f(x)} = E_{f(x)} = \{kx \mid k \in \mathbb{N}\}$.

Per il secondo teorema di ricorsione, esiste $x \in \mathbb{N}$ tale che $\phi_x = \phi_{f(x)}$ e quindi tale che $W_x = E_x = W_{f(x)} = E_{f(x)} = \{kx \mid k \in \mathbb{N}\}$

9.9 Appello 2014-07-02

9.9.1 Esercizio 1

Dimostrare che $A \subseteq \mathbb{N}$ è ricorsivo solo se $A, \overline{A}$ sono RE.

Soluzione

A ricorsivo $\Rightarrow A, \overline{A}$ sono RE.

Essendo A ricorsivo, la sua funzione caratteristica $\mathcal{X}_A$ è calcolabile e può essere utilizzata per definire la funzione semi-caratteristica di A:

$$SC_A(x) = \not\Vdash\Big(\mu w.\overline{sg}\big(\mathcal{X}_A(x)\big)\Big)$$

La funzione semi-caratteristica di $\overline{A}$ può essere definita in modo analogo

$$SC_{\overline{A}}(x) = \not\Vdash\Big(\mu w.\mathcal{X}_A(x)\Big)$$

$A, \overline{A}$ sono RE $\Rightarrow A$ ricorsivo

Entrambe le funzioni semi-caratteristiche sono calcolabili e possono essere combinate per definire la funzione caratteristica.

Sia e_1 un programma tale che $\phi_{e_1} = SC_A$ e e_2 tale che $\phi_{e_2} = \overline{sg}(SC_{\overline{A}})$, entrambi esistono perché le loro funzioni sono calcolabili.

$$\mathcal{X}_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} = \begin{cases} SC_A(x) & x \in A \\ \overline{sg}(SC_{\overline{A}}(x)) & x \notin A \end{cases}$$
$$= \left(\mu w. \left(S\Big(e_1, x, (w)_1, (w)_2\Big) \wedge S\Big(e_2, x, (w)_1, (w)_2\Big) \right) \right)_1$$

9.9.2 Esercizio 2

Definire una funzione $f : \mathbb{N} \to \mathbb{N}$ totale, non calcolabile tale che $f(x) = x/2$ per ogni $x \in \mathbb{N}$ e pari, oppure dimostrare che tale funzione non esiste.

Soluzione

La funzione deve essere totale e non viene specificato nulla per quanto riguarda il valore della funzione sui numeri dispari.

Sia $\{\phi_n\}$ una qualsiasi enumerazione delle funzioni calcolabili.

La funzione f può essere definita come

$$f(x) = \begin{cases} x/2 & x \text{ è pari} \\ \phi_x(x) + 1 & \text{altrimenti} \end{cases}$$

Si ha quindi che la funzione f:

- è totale, perché definita su tutto $\mathbb{N}$
- vale $x/$ se x è pari
- è diversa da tutte le funzioni calcolabili se x è dispari e quindi negli infiniti punti dispari non è calcolabile

9.9.3 Esercizio 3

$$A = \{x | \forall k \in \mathbb{N}. x + k \in W_x\}$$

Soluzione

A sembra non essere RE, perché per provare l'appartenenza è necessario andare a provare infiniti valori di k.

Si può quindi provare la riduzione $\overline{K} \leq_m A$.

Serve quindi una funzione tale che se $x \in \overline{K}$, $f(x)$ sia definita $\forall x' \geq x$ ed una funzione definita su tutto $\mathbb{N}$ soddisfa questa condizione, mentre se $x \in K$, $f(x)$ non deve essere definita in almeno un punto $x' \geq x$.

$$g(x, y) = \begin{cases} 1 & \neg H(x, x, y) \\ \uparrow & H(x, x, y) \end{cases}$$

Non posso usare $x \in \overline{K}$ perché la funzione semi-caratteristica non è calcolabile, devo ragionare sul numero di passi y impiegati dal programma per terminare.

Essendo calcolabile, per SMN esiste f calcolabile e totale, che funziona da funzione di riduzione, perché:

- $x \in \overline{K}$: $\forall y \neg H(x, x, y)$ è vero e quindi $\forall y\, \phi_{f(x)}(y) = 1$ ed in particolare $\phi_{f(x)}(f(x) + k) = 1\, \forall k$, ovvero $f(x) \in A$.
- $x \in K$: $\exists y_0$ tale che $\forall y > y_0$, $\phi_{f(x)}(y) = g(x, y) = \uparrow$, è quindi possibile trovare almeno un valore $y' > y_0$ tale che $y' > x + 0$, pertanto $f(x) \notin A$.

Si ha quindi che A non è RE.

$\overline{A} = \{x | \exists k \in \mathbb{N}. x + k \notin W_x\}$, anche in questo caso sembra non essere RE, perché per verificare l'appartenenza è necessario che il programma x non termini quando riceve in input $x + k$.

Si può quindi provare la riduzione $\overline{K} \leq_m \overline{A}$.

Serve quindi una funzione tale che se $x \in \overline{K}$, $f(x)$ non termini quanto riceve in input $x + k$ e si può osservare che i programmi che calcolano $\emptyset$ sono in $\overline{A}$. Se invece $x \in K$, $f(x)$ deve essere definita su tutti gli input $> x$.

$$g(x, y) = \begin{cases} \uparrow & x \in \overline{K} \\ 1 & x \in K \end{cases} = SC_K(x)$$

Essendo calcolabile, per SMN esiste f calcolabile e totale, che funziona da funzione di riduzione, perché:

- $x \in \overline{K}$: il predicato $\neg H(x, x, y)$ vale $\forall y$ e quindi $\phi_{f(x)}(y) = \uparrow \forall y$, pertanto $f(x) \in \overline{A}$.
- $x \in K$: $\phi_{f(x)}(y) = g(x, y) = 1 \forall y$ ed in particolare $\phi_{f(x)}(y) \downarrow \forall y > x$ e quindi $f(x) \in A$

Segue quindi che anche $\overline{A}$ non è RE.

9.9.4 Esercizio 4

$$V = \{x | E_x \text{ infinito}\}$$

Soluzione

V è saturo, perché $\mathcal{V} = \{f | \, |cod(f)| = \infty\}$.

Banalmente la funzione $id \in \mathcal{V}$ e tutte le sue parti finite non appartengono per definizione a $\mathcal{V}$, quindi per Rice Shapiro V è non RE.

Analogamente $id \notin \overline{\mathcal{V}}$ e la funzione $\emptyset$ appartiene a $\overline{\mathcal{V}}$ ed è parte finita di id, quindi per Rice Shapiro, $\overline{V}$ è non RE.

Sembra troppo facile, forse c'è un trabocchetto

9.9.5 Esercizio 5

Enunciare il secondo teorema di ricorsione e dimostrare che $\exists k | W_k = \{k * i | i \in \mathbb{N}\}$.

Soluzione

Il teorema dice che, data una funzione $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, $\exists e. \phi_e = \phi_{f(e)}$

$$g(x, y) = \begin{cases} 1 & x \text{ è multiplo di } y \\ \uparrow & \text{altrimenti} \end{cases} = \not\Vdash\Big(\mu z. |xz - y|\Big)$$

Essendo g calcolabile, per SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che $g(x, y) = \phi_{f(x)}(y) \forall y$ e quindi si ha che

$$W_{f(x)} = \{x * i | i \in \mathbb{N}\}$$

e per il secondo teorema di ricorsione, esiste x tale che $\phi_x = \phi_{f(x)}$, ovvero $\exists x$ tale che:

$$W_x = W_{f(x)} = \{x * i | i \in \mathbb{N}\}$$

9.10 Appello 2014-03-21

9.10.1 Esercizio 1

Enunciare e dimostrare il teorema di Rice

Soluzione

Il teorema di Rice asserisce che qualsiasi proprietà non banale delle funzioni calcolate dai programmi non è decidibile. In altre parole, sia $A \subseteq \mathbb{N}$ l'insieme che contiene tutti i programmi la cui funzione calcolata gode di una determinata proprietà (ovvero A è un insieme saturo), se la proprietà non è banale ($A \neq \emptyset, \neq \mathbb{N}$) allora A non è ricorsivo.

La non ricorsività si dimostra per riduzione $K \leq_m A$.

Sia e_0 un programma tale che $\phi_{e_0} = \emptyset$. Questo programma può trovarsi in A oppure in $\overline{A}$. Assumiamo che $e_0 \in \overline{A}$.

Essendo A non vuoto, ci sarà almeno un programma $e_1 \in A$ che calcola una qualche funzione.

Si può quindi definire la funzione

$$g(x,y) = \begin{cases} \phi_{e_1}(y) & x \in K \\ \phi_{e_0}(y) = \uparrow & x \notin K \end{cases} = SC_K(x)$$

Tale funzione è calcolabile, perché la funzione semi-caratteristica di K è noto che è calcolabile e quindi per il teorema SMN esiste $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale tale che $\phi_{f(x)}(y) = g(x,y)$.

f è funzione di riduzione perché

- $x \in K$, $\phi_{f(x)}(y) = g(x,y) = \phi_{e_1}(y) \forall y$ e dal momento che A è saturo e che $\phi_{f(x)} = \phi_{e_1}$, $f(x) \in A$.
- $x \notin K$, $\phi_{f(x)} = \emptyset(y) = \phi_{e_0}(y) \forall y$ e dal momento che anche $\overline{A}$ è saturo e che $\phi_{f(x)} = \phi_{e_0}$, $f(x) \in \overline{A}$ e quindi $f(x) \notin A$.

Dal momento che K si riduce ad A, A è non ricorsivo.

Se invece $e_0 \in A$ si può effettuare lo stesso ragionamento, indicando con $\overline{B} = A$ e $B = \overline{A}$. $e_0 \in \overline{B}$ e quindi la dimostrazione è la stessa.

9.10.2 Esercizio 2

Sia $A \subseteq \mathbb{N}$ un insieme e $f : \mathbb{N} \to \mathbb{N}$ una funzione calcolabile. Dimostrare che se A è RE allora $f(A) = \{y \in \mathbb{N} | \exists x \in A. y = f(x)\}$ è RE. Vale anche il contrario?

Soluzione

Se A è RE, la sua funzione semi-caratteristica è calcolabile e quindi esiste un programma e_0 che la calcola e che può essere utilizzato per calcolare la funzione semi-caratteristica di $f(A)$:

$$SC_{f(A)}(x) = \mathbb{1}\left(\mu w.\overline{sg}\left(\underbrace{H\Big(e_0, (w)_1, (w)_2\Big)}_{(w)_1 \in A} \wedge \underbrace{S\Big(e_1, (w)_1, x, (w)_3\Big)}_{f\Big((w)_1\Big) = x}\right)\right)$$

dove e_1 è un programma che calcola f. Devo utilizzare la minimalizzazione illimitata perché f può essere indefinita su qualche punto. Il segno negato serve per azzerare l'espressione quando entrambe le condizioni sono soddisfatte.

Se $f(A)$ è RE la sua funzione semi-caratteristica è calcolabile e può essere utilizzata per definire SC_A:

$$SC_A(x) = SC_{f(A)}(f(x))$$

Perché questo funzioni è necessario che f sia iniettiva e totale, perché sennò possono verificarsi dei casi in cui o $x \in A, x \notin dom(f)$, oppure $fy \in f(A)$ perché $\exists z \in A. f(z) = y, z \neq x$.

Quindi, se f è iniettiva e totale, allora anche A è RE. Altrimenti non è detto che A sia RE.

9.10.3 Esercizio 3

Sia $X \subseteq \mathbb{N}$ un insieme finito, $X \neq \emptyset$ e si definisca $A_X = \{x \in \mathbb{N} : W_x = E_x \cup X\}$. Studiare la ricorsività di A_X.

Soluzione

L'insieme è saturo perché può essere visto come

$$\mathcal{A}_X = \{f \in \mathcal{C} | dom(f) = cod(f) \cup X\}$$

Inoltre, $id \in \mathcal{A}_X$ e $\emptyset \notin \mathcal{A}_X$, ovvero A_X non è banale e quindi per Rice A_X non è ricorsivo.

L'insieme A_X sembra anche essere non-RE, perché per valutare l'appartenenza è necessario esaminare tutti gli elementi del dominio e del codominio.

$$f(x) = \begin{cases} x & x \in X \\ x_0 & \text{altrimenti} \end{cases}$$

con $x_0 \in X$. Questa funzione non appartiene a $\mathcal{A}_X$ perché $dom(f) = \mathbb{N} \neq cod(f) \cup X = X$, mentre la sua parte finita

$$\vartheta(x) = \begin{cases} x & x \in X \\ \uparrow & \text{altrimenti} \end{cases}$$

appartiene ad A_X perché $dom(\vartheta) = cod(\vartheta) = X$. Quindi per Rice-Shapiro A_X è non RE.

Per quanto riguarda il complementare, il ragionamento è simile. $id \notin \overline{\mathcal{A}_X}$ ($dom(id) = cod(id) = \mathbb{N}$), $\emptyset \in \overline{\mathcal{A}_X}$ perché $dom(\emptyset) = \emptyset \neq cod(\emptyset) \cup X$, $\emptyset$ è parte finita di id e quindi per Rice-Shapiro $\overline{\mathcal{A}_X}$ è non RE.

9.10.4 Esercizio 4

Studiare la ricorsività dell'insieme $B = \{x \in \mathbb{N} : \exists k \in \mathbb{N}.k \cdot x \in W_x\}$.

Soluzione

B sembra essere RE:

$$SC_B(x) = \nVdash\Big(\mu w.S\Big(x, (w)_1 \cdot x, (w)_2, (w)_3\Big)\Big)$$

La funzione semi-caratteristica è calcolabile e quindi B è RE.

Probabilmente B non è ricorsivo, $K \leq_m B$.

Se $x \in K$, $f(x) \in B$ e questo è banale dato che B contiene anche le funzioni che sono totali. Se $x \notin K$, $f(x)$ non deve essere in B e quindi non deve esserci un multiplo dell'indice del programma nel dominio della funzione, quindi gli indici dei programmi che calcolano la funzione sempre indefinita non sono in B.

Posso quindi definire

$$g(x, y) = \begin{cases} 1 & x \in K \\ \uparrow & \text{altrimenti} \end{cases} = SC_K(x)$$

g è calcolabile e quindi per il teorema SMN esiste f calcolabile, totale, tale che $\phi_{f(x)} = g(x, y)$ e che può essere utilizzata come funzione di riduzione perché

- $x \in K$: $\phi_{f(x)}(y) = g(x, y) = 1 \forall y$, ovvero $W_{f(x)} = \mathbb{N}$ e quindi di sicuro $\exists k \in \mathbb{N}.k \cdot f(x) \in W_{f(x)}$
- $x \notin K$: $\phi_{f(x)}(y) = g(x, y) = \uparrow \forall y$ ovvero $W_{f(x)} = \emptyset$ e quindi di sicuro $\nexists k \in \mathbb{N}.k \cdot f(x) \in W_{f(x)}$.

Quindi B non è ricorsivo ed è RE. $\overline{B}$ è per forza non RE.

9.10.5 Esercizio 5

Enunciare il secondo teorema di ricorsione. Dimostrare che $B = \{x \in \mathbb{N} : \exists k \in \mathbb{N}.k \cdot x \in W_x\}$ non è saturo.

Soluzione

Sia $f : \mathbb{N} \to \mathbb{N}$ calcolabile e totale, allora $\exists e.\phi_e = \phi_{f(e)}$.

$$g(x,y) = \begin{cases} 1 & y = x \\ \uparrow & \text{altrimenti} \end{cases} = \not\Vdash\Big(\mu k.|x-y|\Big)$$

Essendo g calcolabile, posso applicare SMN per trovare $\phi_{f(x)}(y) = g(x,y)$ e per il secondo teorema di ricorsione si ha che:

$$\exists e \text{ tale che} \phi_e(y) = \phi_{f(e)}(y) = g(e,y) = \begin{cases} 1 & y = e \\ \uparrow & \text{altrimenti} \end{cases}$$

e quindi tale che $W_e = W_{f(e)} = \{e\}$ e quindi $e \in B$ per $k = 1$.

Dal momento che ϕ_e è calcolabile, ci sono infiniti indici che la calcolano e di sicuro ci sono degli indici $e' > e$. Ovvero tali che $W_{e'} = W_e = e$.

Essendo $e' > e$, $\forall k \in \mathbb{N}, k \cdot e' \notin W_{e'} = W_e$ e quindi e' non è in B, ovvero B non è saturo.

C'è un caso particolare se $e = 0$, perché in quel caso tutti gli e' sono multipli per $k = 0$.

Per gestire ciò si può forzare il fatto che il punto fisso sia $\neq 0$, ovvero si sceglie un e_0 tale che $\phi_{e_0} = \phi_0$ e si riapplica il secondo teorema di ricorsione utilizzando come funzione:

$$f'(x) = \begin{cases} f(x) = e & f(x) \neq 0 \\ e_0 & \text{altrimenti} \end{cases}$$

9.11 Appello 2014-04-03

9.11.1 Esercizio 1

Dimostrare il teorema di struttura dei predicati semidecidibili.

Soluzione

Se $Q(x,y)$ è decidibile, allora la sua funzione caratteristica $\mathcal{X}_Q$ è calcolabile e può essere utilizzata per definire quella semi-caratteristica di P.

$$SC_P(x) = \begin{cases} 1 & \exists y.Q(x,y) \\ \uparrow & \text{altrimenti} \end{cases} = \not\Vdash\left(\mu w.\overline{sg}\Big(\mathcal{X}_Q(x,w)\Big)\right)$$

Se invece P è semi-decidibile, allora esiste un programma e che calcola la sua funzione semi-caratteristica. Inoltre, se P è vero, esiste un numero di passi che fanno terminare il programma e:

$$P(x) \equiv \mu y.H(e,x,y)$$

Indicando con $Q(x,y)$ il predicato $H(e,x,y)$ si ottiene il predicato decidibile desiderato.

9.11.2 Esercizio 2

Sia $A \subseteq \mathbb{N}$ e sia $f : \mathbb{N} \to \mathbb{N}$ una funzione calcolabile. Dimostrare che se A è ricorsivo allora $f^{-1}(A) = \{x \in \mathbb{N} | f(x) \in A\}$ è RE. L'insieme $f^{-1}(A)$ è anche ricorsivo?

Soluzione

Sia e un programma che calcola f. Questo programma può essere utilizzato per definire la funzione semi-caratteristica di $f^{-1}(A)$:

$$SC_{f^{-1}(A)}(x) = \begin{cases} 1 & x \in f^{-1}(A) \\ \uparrow & \text{altrimenti} \end{cases} = \begin{cases} 1 & f(x) \in A \\ \uparrow & f(x) \notin A \end{cases} = \nVdash \left(\mu w. \overline{sg} \Big(S\Big(e, x, (w)_1, (w)_2\Big) \wedge \mathcal{X}_A\Big((w)_1\Big) \Big) \right)$$

Questa funzione è calcolabile perché definita utilizzando funzioni calcolabili, e quindi $f^{-1}(A)$ è RE.

Dal momento che non è garantito che f è totale, l'insieme $f^{-1}(A)$ non è ricorsivo, perché può esistere un x per il quale $f(x)$ non è definita e quindi non può essere determinato se $f(x)$ è in A o meno.

9.11.3 Esercizio 3

$$A = \{x : W_x \cap E_x \neq \emptyset\}$$

Soluzione

L'insieme è saturo, perché contiene tutti i programmi le cui funzioni calcolate hanno $dom(f) \cap cod(f) \neq \emptyset$.

$$\mathcal{A} = \{f \in \mathcal{C} : dom(f) \cap cod(f) \neq \emptyset\}$$

Quindi per il teorema di Rice A non è ricorsivo.

A sembra essere ricorsivo, perché una volta trovato un punto in comune tra il dominio e il codominio, il test di appartenenza può terminare.

$$SC_A(x) = \nVdash \left(\mu w. \overline{sg} \Big(S\big(x, (w)_1, (w)_1, (w)_2\big) \Big) \right)$$

Essendo la funzione semi-caratteristica calcolabile, A è RE e non ricorsivo per quanto detto prima. $\overline{A}$ deve quindi essere non RE.

Dimostrazione che $\overline{A}$ è non RE $\overline{\mathcal{A}}$ probabilmente è non RE, perché per determinare l'appartenenza è necessario valutare tutto il dominio e il codominio per determinare se l'intersezione è vuota.

La funzione $f(x) = 1$ è in $\overline{A}$ perché $dom(f) \cap cod(f) = \{1\}$ e quindi $f \notin \overline{\mathcal{A}}$.

La sua parte finita:

$$\vartheta(x) = \begin{cases} 1 & x = 0 \\ \uparrow & \text{altrimenti} \end{cases}$$

è in $\overline{\mathcal{A}}$ perché $dom(\vartheta) \cap cod(\vartheta) = \emptyset$. Quindi per Rice-Shapiro $\overline{A}$ è non RE.

9.11.4 Esercizio 4

$$B = \{x \in \mathbb{N} : \forall k \in \mathbb{N}. k + x \in W_x\}$$

Soluzione

B sembra essere non-RE perché per decidere l'appartenenza bisogna controllare infiniti valori.

Si va quindi di riduzione $\overline{K} \leq_m B$.

$$g(x,y) = \begin{cases} 1 & \neg H(x,x,y) \\ \uparrow & H(x,x,y) \end{cases} = \not\Vdash\Big(\mu w.H(x,x,y)\Big)$$

È calcolabile, quindi per SMN esiste f che fa da funzione di riduzione:

- $x \in \overline{K}$: $\phi_{f(x)}(y) = g(x,y) = 1 \forall y$. $\phi_{f(x)}(y)$ è quindi sempre definita ed in particolare $\forall k \in \mathbb{N}, f(x)+k \in W_{f(x)}$ e quindi $f(x) \in B$.
- $x \in K$: $\exists y_0 \in \mathbb{N}.\phi_{f(x)}(y) = g(x,y) = \uparrow \forall y > y_0$. Quindi $\exists k \in \mathbb{N}.f(x)+k \notin W_{f(x)}$ e quindi $f(x) \notin B$.

Si ha quindi che B è non-RE.

$\overline{B} = \{x : \exists k \in \mathbb{N}.x+k \notin W_x\}$ sembra essere non-RE perché per trovare un k che soddisfa la condizione di appartenenza è necessario aspettare la terminazione del programma su un input in cui non termina.

$$g(x,y) = \begin{cases} \uparrow & x \in \overline{K} \\ 1 & x \in K \end{cases} = SC_K(x)$$

Per SMN esiste f:

- $x \in \overline{K}$: $\phi_{f(x)}(y) = g(x,y) = \uparrow \forall y$ e quindi $f(x) \in \overline{B}$.
- $x \in K$: $\phi_{f(x)}(y) = g(x,y) = 1 \forall y$, in particolare $W_{f(x)} = \mathbb{N}$ e quindi $f(x) \in B$.

$\overline{B}$ quindi è non-RE.

9.11.5 Esercizio 5

Dimostrare che l'insieme non è saturo:

$$C = \{x : \phi_x(x) = x^2\}$$

Soluzione

Il secondo teorema dice che data f calcolabile e totale, esiste e tale che $\phi_e = \phi_{f(e)}$.

$$g(x,y) = \begin{cases} y^2 & y = x \\ \uparrow & \text{altrimeniti} \end{cases} = y^2 \cdot \not\Vdash\Big(\mu w.|y-x|\Big)$$

Per SMN esiste f. $\phi_{f(x)}(x) = g(x,x) = x^2 \to f(x) \in C$.
Per il secondo teorema $\exists e.\phi_e(e) = \phi_{f(e)}(e) = g(e,e) = e^2 \to e \in C, W_e = \{e^2\}$.
Sia $e' \in \mathbb{N} : \phi_{e'} = \phi_e, e' \neq e \to \phi_{e'}(e') = \uparrow \neq e'^2 \to e' \notin C$.
Ma dato che $\phi_e = \phi_{e'}$ si ha che l'insieme C non è saturo.