12ª Lezione

Continuiamo la nostra panoramica sugli insiemi decidibili e sugli insiemi riducibili.

12.1 Riducibilita' di A a k, con A semidecidibile

Vediamo ora di dimostrare che, **dato un qualunque insieme semidecidibile A, $A \leq_T k$**. Dal momento che A e' semidecidibile, allora la sua funzione semicaratteristica e' calcolabile. Vediamo tale funzione:

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

Questa funzione non e' totale, quindi non puo' essere la nostra trasformazione s(x) che usiamo per passare da A a k. Per renderla totale, ci aggiungiamo una variabile per poter poi, cosi', applicare il teorema del parametro. Otteniamo quindi la seguente funzione:

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

Essendo $X_A(x)$ calcolabile, allora anche f(x,y) e' calcolabile. Possiamo quindi applicare ad essa il teorema del parametro. Esiste, quindi, una funzione calcolabile totale S(x) tale che $\Phi_{S(x)}(y)=f(x,y)$. Da cio' deduciamo che:

- $x \notin A \Rightarrow \Phi_{S(x)}(y)=\uparrow\ (\forall y) \Rightarrow \mathbf{S(x) \notin k}$ — In questo caso S(x) e' un programma per la funzione vuota. Quindi, S(x) non termina su nessun input (y), tanto meno su se stesso.
- $x \in A \Rightarrow \Phi_{S(x)}(y)=1\ (\forall y) \Rightarrow \mathbf{S(x) \in k}$ — In questo caso S(x) e' un programma per una funzione costante totale. Quindi, S(x) termina su tutti gli input (y), percio' anche su se stesso.

Questo indica che S(x) e' proprio la traformazione di riducibilita' s(x) che cercavamo, quindi $\mathbf{A \leq_T k}$.

Da questa dimostrazione possiamo dedurre che k e' l'insieme semidecidibile piu' complesso, in quanto qualsiasi altro insieme semidecidibile puo' essere ridotto a k. In particolare, anche ogni insieme decidibile e' riducibile a k, in quanto un insieme decidibile e', come sappiamo, anche semidecidibile.

12.2 Esercizi sulla decidibilita' e riducibilita' di insiemi

Vediamo ora un'altra serie di esercizi che ha come denominatore comune il concetto di riducibilita', oltre a quello di decidibilita' e semidecidibilita'.

1. Testo:
Dato l'insieme $I=\{x : (\forall y)\ P_x\downarrow x$ in un numero di passi di computazione $\geq y+1\}$, determinare se esso e' semidecidibile.

Svolgimento:
I e' l'insieme degli input che, applicati a qualsiasi programma, lo fanno terminare nel numero di passi definito sopra. Ma, dal momento che esistono programmi che non terminano per nessun input (ad esempio i programmi per la funzione vuota), non vi sara' mai un input che faccia terminare tutti I programmi e quindi, per forza di cosa, I sara' l'insieme vuoto. Ovviamente l'insieme vuoto e' decidibile (e' facile fare un programma che mi dice se un input sta nell'insieme vuoto) e quindi e' anche semidecidibile.

2. Testo:
Dato l'insieme $J=\{x : (\exists z)(\forall y)\ P_z\downarrow y$ in un numero di passi di computazione $\leq x\}$, determinare se esso e' decidibile.

Svolgimento:
Consideriamo il programma P_z =**X1:=0**. Questo programma, composto da una istruzione, sara' tale per cui, per ogni input, la computazione termina in un numero di passi $\leq x$, dal momento che il numero di passi sara' sempre 1 e che non consideriamo programmi lunghi zero istruzioni. Quindi **J=N\{0}**.
Ora, dobbiamo provare che l'insieme N\{0} e' decidibile. Questo e' vero perche' {0} e' decidibile (un programma che lo decide potrebbe essere " **IF x=0 THEN** $x\in\{0\}$ **ELSE** $x\notin\{0\}$ "), per cui anche il complementare di {0}, cioe' J, e' decidibile.

3. Testo:
Dato l'insieme $A=\{(x,y) : \Phi_x(z) = \Phi_y(z)$ per ogni input $z\}$, determinare se esso e' decidibile.

Svolgimento:
In pratica A e' l'insieme delle coppie di programmi equivalenti. Supponiamo, per assurdo, che A sia decidibile e consideriamo l'insieme $B=\{x : (x,C_0) \in A\}$ dove C_0 = **"1 Goto 1"**, ovvero l'insieme dei programmi equivalenti alla funzione nulla (qui C_0 e' uno dei possibili programmi per la funzione nulla). Dal momento che abbiamo supposto A decidibile, allora esistera' un programma che calcola la sua funzione caratteristica. Ma allora, in questo programma bastera' porre $y = C_0$ e avremo trovato un programma che decide B, da cui deduciamo che B e' decidibile. Ora, se proviamo a scrivere !B, ci accorgiamo che questo e' l'insieme dei programmi che terminano su almeno un input, che avevamo visto essere semidecidibile ma non decidibile. Ma se !B e' non decidibile, allora anche B deve essere non decidibile, e cio' ci porta ad una contraddizione, da cui deduciamo che **A non e' decidibile.**
Allo stesso modo possiamo ragionare per dimostrare che A non e' semidecidibile. Supponiamo per assurdo che A sia semidecidibile. Ma allora anche B sara' semidecidibile (sempre grazie alla sostituzione di y con C_0 nel programma che semidecide A). Ma sappiamo gia' che !B e' semidecidibile mentre B non e' decidibile, quindi B non e' semidecidibile, e cosi' otteniamo di nuovo una contraddizione che ci porta a dire che **A non e' semidecidibile.**

12.3 Esercizi su argomenti vari

Vediamo ora due esercizi su argomenti non attinenti a queste ultime lezioni. Uno riguarda il paradigma funzionale, l'altro la macchina di Touring:

1. Testo:
Scrivere un programma funzionale iterativo che calcola la seguente funzione g(x):

$$g(x)=\begin{cases}5 & \text{se } x=0\\ 2x+1 & \text{se } x\neq 0\end{cases}$$

Svolgimento:
Utilizzando le funzioni Sg e !Sg, la soluzione del problema diventa agevole. In linea di massima, il nostro programma P sara' del tipo:

$$\boxed{P(x) = 5*(!Sg(x)) + (2x+1)Sg(x)}$$

Il che porta, partendo dall'input x, alle seguenti trasformazioni.

$$\boxed{x \rightarrow 5, !Sg(x), 2x+1, Sg(x) \rightarrow 5*(!Sg(x)), (2x+1)*Sg(x) \rightarrow 5*(!Sg(x)) + (2x+1)Sg(x)}$$

Appare chiaro che ci servira' un sottoprogramma per calcolare il numero 5, ed un sottoprogramma per calcolare 2x+1. Questi saranno:

- $\underline{5} =_{def}$ **(0; S; S; S; S; S)**
- **h(x)** $=_{def}$ **(D; +; S)**

Il programma finale sara' quindi:

$$P(x) = (P^2_{2,1} ; 5)\hat{}(!Sg)\hat{}h \hat{}Sg) ; (* \| *) ; +$$

2. Testo:

Scrivere un programma per una macchina di Touring monodimensionale, con alfabeto A = {0, blank} che termina la computazione su un numero pari di zeri del nostro input (per convenzione zero zeri e' un numero pari di zeri).

Svolgimento:

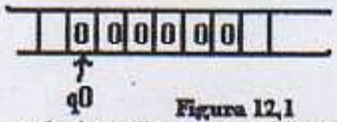

Figura 12.1

La soluzione di questo problema e' molto semplice. Poniamo :

- **q0** = stato che corrisponde ad un numero pari di zeri letti
- **q1** = stato che corrisponde ad un numero dispari di zeri letti

Detto questo, il nostro programma per la macchina di Touring sara' composto dalle seguenti tre righe:

Configurazione da cui parto		**Azioni che commetto e stato cui vado**		
Stato	**Cosa leggo**	**Cosa scrivo**	**Nuovo stato**	**Direzione testina**
q_0	0	0	q_1	D
q_1	0	0	q_0	D
q_1	blank	blank	q_1	D

13ª Lezione

Nell'ambito della decidibilita' e riducibilita' di insiemi, vediamo adesso altri due esempi che ci chiariranno le idee prima dello studio dei due importanti teoremi di Rice.

13.1 Decidibilita' dell'insieme dei programmi che terminano sull'input 0

Consideriamo l'insieme **I = { i | Pi termina la computazione nell'input 0}**. Si puo' facilmente dimostrare che **I e' semidecidibile**. Questo perche', se prendiamo un qualsiasi $i \in N$, e lo codifichiamo nel corrispondente programma Pi, potremo lanciare tale programma Pi sull'input 0. A questo punto, se $Pi\downarrow 0$ allora $i \in I$, altrimenti si ha che $Pi\uparrow 0$ e si entra in ciclo: abbiamo cosi' costruito un programma che semidecide I.

Vediamo ora di dimostrare che **I non e' decidibile**, e quindi che **!I non e' semidecidibile**. A questo scopo, proviamo a ridurre k ad I. Sappiamo che k e' semidecidibile, per cui esiste almeno una funzione che semidecide k (ne abbiamo gia' viste un paio nelle scorse lezioni). Consideriamo ora la seguente funzione calcolabile a due variabili che semidecide k:

$$f(x,y) = \begin{cases} y & \text{se } x \in k \\ \uparrow & \text{altrimenti} \end{cases}$$

Poiche' k e' semidecidibile, questa funzione e' chiaramente calcolabile, per cui possiamo applicare ad essa il teorema del parametro. Esistera' quindi una funzione totale calcolabile S(x) tale che $\Phi_{S(x)}(y) = f(x,y)$. Cio' comporta che:

- $x \in K \Rightarrow \Phi_{S(x)}(y)=y \Rightarrow S(x) \in I$ Quindi in questo caso S(x) e' un programma che calcola la funzione identica ed e' definito su tutti gli y, in particolare per y=0.
- $x \notin K \Rightarrow \Phi_{S(x)}(y)=\uparrow \Rightarrow S(x) \notin I$ Quindi in questo caso S(x) e' un programma che calcola la funzione vuota, per cui avremo che $S(x)\uparrow 0$.

Quindi abbiamo trovato la nostra trasformazione S(x) che riduce k ad I, per cui $k \leq_T I$. Ricordando il teorema del paragrafo 11.2, possiamo concludere che essendo k non decidibile, allora anche I non e' decidibile. Da questo deriva che !I non e' semidecidibile.

13.2 Decidibilita' dell'insieme dei programmi che calcolano una funzione f

Consideriamo l'insieme $I = \{x \mid \Phi_x = f\}$ e ci chiediamo se questo sia o no decidibile. Ovviamente, questo problema non e' ben determinato fino a quando non si chiarisca di quale f stiamo parlando. In generale, infatti, avremo un insieme I diverso per ogni scelta diversa di f.

Prima di tutto, consideriamo il caso in cui $f = f_\phi$, cioe' il caso in cui f sia la funzione nulla. In questa circostanza l'insieme I non e' ne' decidibile, ne' semidecidibile, in quanto non e' possibile sapere con certezza se un programma non termina mai la computazione su nessun input. In questo caso, pero' !I sara' semidecidibile, in quanto

abbiamo visto che con l'algoritmo illustrato dalla tabella di figura 11.1 e' possibile decidere l'insieme dei programmi che terminano su almeno un input, cioe' !I.

Vediamo ora il caso in cui $f \neq f_\phi$. In questa circostanza, cerchiamo di dimostrare la non decidibilita' di I provando a ridurre k ad I. A questo scopo, prendiamo la seguente funzione h(x,y):

$$h(x,y)=\begin{cases} f(y) & \text{se } x \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Questa funzione puo' essere calcolata da una funzione che semidecide k, e puo' entrare in ciclo solo quando $x \notin K$ oppure qualora $x \in K$ ed $f\uparrow y$. Dunque, dal momento che h(x,y) e' calcolabile, posso applicare ad essa il teorema del parametro. Esistera' dunque una funzione calcolabile e totale S(x) tale che $\Phi_{S(x)}(y) = h(x,y)$. In questo caso, avremo che:

- $x \in K \Rightarrow \Phi_{S(x)}(y)=f(y) \Rightarrow S(x) \in I$ — In questo caso S(x) e' un programma che calcola la funzione f, in quanto vediamo che agli stessi input (y) corrispondono gli stessi output.
- $x \notin K \Rightarrow \Phi_{S(x)}(y)=\uparrow \Rightarrow S(x) \notin I$ — In questo caso S(x) e' un programma che calcola la funzione vuota, non f, per cui, per forza di cose, S(x) non sta in I.

Abbiamo quindi dimostrato che S(x) e' la trasformazione che ci permette di ridurre k ad I ($k \leq_T I$). Cio' ci permette di affermare che **I non e' decidibile**.

13.3 Insiemi che rispettano le funzioni

Diciamo che **un insieme I rispetta le funzioni** se, $\forall x \in I, (\Phi_x = \Phi_y \Rightarrow y \in I)$, cioe', preso un programma da un insieme di programmi, esso sara' equivalente ad un altro programma solo se quest'ultimo fa anch'essa parte dello stesso insieme di programmi.

Notiamo che, **se I rispetta le funzioni, allora anche !I rispetta le funzioni**. Infatti, se supponiamo per assurdo che !I non rispetti le funzioni, allora ci troveremmo con due programmi equivalenti che stanno uno in I, e uno in !I, per cui anche I non rispetterebbe le funzioni, che e' una contraddizione alla nostra ipotesi.

Facciamo qualche esempio di insiemi che rispettano o non rispettano le funzioni:

a) $I = \{x : \Phi_x = f\}$ Questo insieme rispetta le funzioni, in quanto, preso un programma x che calcola f, un altro programma y sara' equivalente a x solo se calcolera' anch'esso f.

b) $I = \{ x : \Phi_x\downarrow y$ **per almeno un** $y\}$ Questo insieme rispetta le funzioni, in quanto, preso un programma x che termina su almeno un input, un altro programma z sara' equivalente ad x solo se terminera' su almeno un input.

c) $I = \{2, 5, 100\}$ Questo insieme non rispetta le funzioni, in quanto se aggiungiamo, ad esempio, al programma numero 2 delle istruzioni che non fanno nulla, otterremo un programma equivalente al 2, che pero' con ogni probabilita' non avra' codifica in I.

d) $K=\{x : P_x\downarrow x\}$ Questo insieme non rispetta le funzioni. Lo dimostreremo piu' avanti. Comunque, intuitivamente, gia' si vede come, se aggiungiamo istruzioni inutili ad un programma che termina su se stesso, esso cambia codifica, ma, in generale, non e' detto che termini ancora su se stesso.

13.4 Primo teorema di Rice

Vediamo adesso il primo di una serie di teoremi che ci dara' una grossa mano nel dimostrare la decidibilita' o meno di certi insiemi.

1° teorema di Rice:

Sia $I \subseteq N$ un insieme che rispetta le funzioni. Allora

1. *I e' decidibile $\Leftrightarrow I=\phi$ o $I=N$*
2. *Se $\{x : \Phi_x = f_\phi\} \subseteq I \neq N$ allora I non e' semidecidibile*
3. *Se $\{x : \Phi_x = f_\phi\} \subseteq !I \neq N$ allora !I non e' semidecidibile*

Dim (1. $\Leftarrow$):

Banale, perche' se $I=\phi$ o $I=N$ allora e' facile trovare un programma che decida I, quindi I e' decidibile.

Dim (1,←):
Consideriamo la situazione in cui I≠ϕ e I≠N. Sappiamo che, dato che I rispetta le funzioni, allora anche !I rispetta le funzioni. Dunque I programmi che calcolano la funzione vuota, che sono tutti equivalenti fra loro, stanno tutti in I o tutti in !I. Consideriamo la situazione in cui tutti I programmi per la ~~stringa~~ funzione vuota stiano in I. Ci troveremmo allora di fronte alla situazione di figura 13.1:

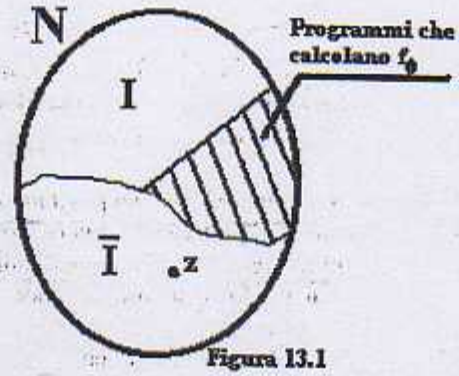

Figura 13.1

Consideriamo ora un certo z∈!I (come mostrato in figura) e costruiamo la seguente funzione in due variabili che semidecide k:

$$f(x,y) = \begin{cases} \phi_z(y) & \text{se } x \in k \\ \uparrow & \text{altrimenti} \end{cases}$$

Questa funzione, come del resto h(x,y) del paragrafo 13.2, e' chiaramente calcolabile, in quanto ,al massimo, il programma che la calcola potrebbe entrare in ciclo se una certa y non fosse definita in Pz, ma questo in fondo non ci deve interessare (intanto il programma l'abbiamo scritto). Possiamo quindi applicare a f(x,y) il teorema del parametro. Esiste percio' una funzione S(x) totale e calcolabile tale che $\Phi_{S(x)}(y) = f(x,y)$. Da questo ricaviamo che:

- $x \in K \Rightarrow \Phi_{S(x)}(y) = \Phi_z(y) \Rightarrow S(x) \notin I$ — In questo caso S(x) e' un programma equivalente a Pz. Dal momento che Pz sta in !I e che !I rispetta le funzioni, allora anche sta in !I.
- $x \notin K \Rightarrow \Phi_{S(x)}(y) = \uparrow \Rightarrow S(x) \in I$ — In questo caso S(x) e' un programma che calcola la funzione vuota. Dal momento che tutti I programmi che la calcolano stanno in I, ed I rispetta le funzioni, allora anche S(x) sta in I.

Con questo abbiamo dimostrato che S(x) e' la trasformazione che ci permette di ridurre k a !I ($k \leq_T$!I). Quindi, dal momento che k non e' decidibile, anche !I non e' decidibile, quindi I non e' decidibile.
Abbiamo quindi dimostrato che **(I≠ϕ e I≠N) ⇒ I non decidibile**, quindi **I decidibile ⇒ (I=ϕ o I=N).**

Dim (2):
Se l'insieme dei programmi che calcolano la funzione vuota sta in I, allora abbiamo visto che $k \leq_T$!I, da cui per le proprieta' dell'operatore di riduzione, $!k \leq_T I$, e dal momento che !k non e' semidecidibile, allora **I non e' semidecidibile.**

Dim (3):
Se l'insieme dei programmi che calcolano la funzione vuota sta in !I, allora si effettua la dimostrazione sopra al contrario (si prende z∈I, ecc) ricavando che $k \leq_T I$, da cui per le proprieta' dell'operatore di riduzione, $!k \leq_T$!I, e dal momento che !k non e' semidecidibile, allora **!I non e' semidecidibile.**

13.5 Decidibilita' dell'insieme di programmi che calcolano funzioni iniettive

Consideriamo l'insieme I ={x : Φ_x e' iniettiva} dei programmi che calcolano funzioni iniettive. Questo insieme rispetta le funzioni, in quanto un programma equivalente ad uno che calcola una funzione iniettiva, calcolera' anch'esso una funzione iniettiva. Inoltre (I≠ϕ e I≠N) in quanto esistono programmi che calcolano funzioni iniettive, ma non sono tutti I programmi esistenti. Quindi possiamo applicare il primo teorema di Rice e, dal momento che la funzione vuota e' iniettiva, $f_\phi \in I$. Da questo ricaviamo che **I non e' semidecidibile.**

14ª Lezione

In questo capitolo vedremo il secondo teorema di Rice, grazie al quale riusciremo a risolvere problemi non altrimenti risolubili col primo teorema. Partiremo enunciando e dimostrando tale teorema, per poi passare a vedere esempi di insiemi cui questo problema e' applicabile.

Prima di tutto questo, pero', vediamo un concetto che ci sara' utile in seguito.

14.1 Relazione d'ordine parziale fra funzioni

Esponiamo in questo paragrafo un concetto di relazione d'ordine parziale riferito a funzioni. Tale relazione d'ordine viene denotata col simbolo $\leq$, e, date due funzioni f e g, e' cosi' definita:

$$f \leq g \text{ se: } \begin{cases} \text{dom } f \subseteq \text{dom } g \\ x \in \text{dom } f \rightarrow f(x) = g(x) \end{cases}$$

Figura 14.1

In pratica, se $f \leq g$, la funzione g rappresenta una **estensione** della funzione f, in quanto e' uguale ad f in tutto e per tutto quando consideriamo input definiti anche su f. L'unica differenza tra le due funzioni e' che g e' definita, in generale, anche per qualche valore in cui f non e' definita.

Dal momento che abbiamo instaurato una relazione d'ordine tra funzioni, appare naturale chiedersi quale sia la funzione in assoluto piu' "piccola" tra tutte quelle esistenti in natura, e quale sia invece la piu' "grande" (sempre che ne esista una che risponda a tale requisito).

E' chiaro che la piu' piccola funzione, rispetto a tale relazione d'ordine, e' la funzione vuota f_ϕ, in quanto essa, non terminando su nessun input, non puo' estendere nessun'altra funzione (a parte, se vogliamo, se stessa), mentre puo' essere estesa da qualsiasi funzione. Le funzioni immediatamente piu' grandi della funzione vuota sono quelle che terminano su un solo input (chiamiamole g_i). Le funzioni immediatamente piu' grandi delle g_i sono quelle che terminano su due input, uno dei quali deve essere quello su cui e' definita la funzione g_i (per il quale avranno anche uguale output), ecc.

Contrariamente a molte relazioni d'ordine, questa contempla anche una classe di "funzioni massime" che non possono essere estese da nessun'altra. Queste sono le funzioni totali, che hanno cioe' come dominio tutto N.

14.2 Secondo teorema di Rice

2° teorema di Rice:

Sia $I \subseteq N$ un insieme che rispetta le funzioni. Supponiamo che esistano due funzioni calcolabile f e g tali che:

1. $\{x : \Phi_x = f\} \subseteq I$
2. $\{x : \Phi_x = g\} \subseteq \overline{I}$
3. $f \leq g$

allora I non e' semidecidibile

Dim:
Supponiamo di avere un insieme I e due funzioni calcolabili f e g che soddisfano alle condizioni iniziali del secondo teorema di Rice. La nostra idea e di riuscire a ridurre !k ad I, in modo da dimostrare che I non e' semidecidibile. A questo scopo, prendiamo in considerazione la seguente funzione:

$$h(x,y)=\begin{cases} g(y) & \text{se } x\in k \\ f(y) & \text{altrimenti} \end{cases}$$

Come al solito il nostro scopo e' di riuscire ad applicare il teorema del parametro ad h(x,y), per cui, la prima cosa da fare, e' dimostrare che h(x,y) e' calcolabile. Per farlo, facciamo vedere che esiste un programma che la calcola. Questo programma, fa partire in parallelo un programma che semidecide k (con input x) e un programma che calcola f (con input y). A questo punto, ci possiamo trovare di fronte a tre situazioni:

- Il programma che semidecide k termina per primo, quindi $x\in k$. In questo caso, faccio subito partire il programma che calcola g (con input y). Il risultato di questa computazione sara' h(x,y).
- Il programma che calcola f(y) termina per primo. In questo caso, non ci interessa scoprire se x stia o non stia in k, in quanto il nostro risultato della computazione h(x,y) e' sempre f(y). Questo perche', dal momento che $f \leq g$, per questo particolare valore di y in cui f e' definita abbiamo che f(y)=g(y), per cui, sia che x stia in k, sia che x non stia in k, il valore di h(x,y) e' sempre lo stesso.
- Nessuno dei due programmi termina. In questo caso ci troviamo di fronte alla situazione in cui $x\notin k$ e $f\uparrow y$. In questa situazione e' perfettamente legittimo che h(x,y) cicli all'infinito, in quanto sta cercando un valore per f(y) che non potra' mai trovare.

Ora, avendo trovato un programma per h(x,y), abbiamo dimostrato che h(x,y) e' calcolabile. Possiamo quindi applicare ad essa il teorema del parametro, per cui esistera' una funzione totale e calcolabile S(x) tale che $\Phi_{S(x)}(y) = h(x,y)$. In questo caso, avremo che:

- $x\in k \Rightarrow \Phi_{S(x)}(y)=g(y) \Rightarrow S(x)\notin I$ — Qui S(x) e' un programma che calcola g. Dal momento che !I rispetta le funzioni e g sta in !I, allora anche S(x) sta in !I.
- $x\notin k \Rightarrow \Phi_{S(x)}(y)=f(y) \Rightarrow S(x)\in I$ — Qui S(x) e' un programma che calcola f. Dal momento che I rispetta le funzioni ed f sta in I, allora anche S(x) sta in I.

Abbiamo quindi dimostrato che S(x) e' la trasformazione che ci permette di ridurre k a !I. Quindi, per le proprieta' dell'operatore di riduzione, $!k \leq_T I$, da cui, dal momento che !k non e' semidecidibile deduciamo che **I non e' semidecidibile**.

Notiamo che in questo caso siamo riusciti a calcolare la funzione h(x,y) nonostante k non sia decidibile.

14.3 Qualche esempio di applicazione del secondo teorema di Rice

Vediamo due esempi di applicazione del secondo teorema di Rice che ci chiariranno ancor meglio le idee sul significato di questo teorema.

14.3.1 Decidibilita' dell'insieme dei programmi che calcolano f_ϕ

Consideriamo l'insieme I dei programmi che calcolano la funzione vuota. Ovviamente I rispetta le funzioni perche' un programma che calcola f_ϕ e' equivalente ad un altro programma che calcola f_ϕ. Inoltre, $I \neq \phi \neq N$, in quanto esistono programmi per la funzione vuota, ma non sono certo tutti quelli esistenti.

Ora, se prendiamo

- $f(x) = f_\phi(x)$ (funzione vuota)
- $g(x) = k$ (funzione costante)

abbiamo che f sta in I, mentre g sta in !I, mentre e' banale constatare che $f \leq g$. Quindi possiamo applicare il secondo teorema di Rice, ottenendo come risultato che **I non e' semidecidibile**.

14.3.2 Decidibilita' di un insieme di programmi che calcola funzioni "semi - identiche"

Consideriamo il seguente insieme:

$$I=\{x \mid \phi_x(y)=\begin{cases} y & \text{se y e' pari} \\ \uparrow & \text{altrimenti} \end{cases}\}$$

Questo insieme rispetta le funzioni, in quanto e' l'insieme di tutte le funzioni equivalenti di questo tipo. Inoltre, $I \neq \phi \neq N$, in quanto non tutti I programmi calcolano la funzione sopra, ma sicuramente ve n'e' una che la calcola (un programma che la calcola sara' un programma che semidecide l'insieme dei numeri pari).

Quindi possiamo applicare il primo teorema di Rice, il quale ci dice subito che **I non e' decidibile**. Inoltre, dal momento che la funzione vuota sta in !I, deduciamo che **!I non e' semidecidibile**.

Ora, consideriamo le seguenti due funzioni:

- $f(y)=\begin{cases} y & \text{se y e' pari} \\ \uparrow & \text{altrimenti} \end{cases}$
- **g(y) = y (funzione identica)**

E' chiaro che $f \leq g$, in quanto g e' una estensione di f ai numeri dispari. E' anche chiaro che f sta in I, mentre g, non essendo equivalente ad f, sta in !I. Possiamo quindi applicare il secondo teorema di Rice e concludere **che I e' non semidecidibile**, il che' conclude il nostro studio della funzione.

15ª Lezione

In questo capitolo verra' illustrato il terzo ed ultimo teorema di Rice, il quale ci permettera' di individuare le caratteristiche di alcuni insiemi, che altrimenti, con I primi due teoremi, sarebbero rimaste nascoste.

15.1 Restrizione finita di una funzione

Prima di enunciare e dimostrare il terzo teorema di Rice, chiariamo un concetto che verra' richiamato con frequenza nel corso dei prossimi paragrafi.

Data una certa funzione f, chiameremo **restrizione finita di f** una certa funzione g, con g < f, tale che g ha **dominio finito**. In pratica, a f puo' essere associato un numero di restrizioni finite pari alla cordinalita' dell'insieme delle parti del suo dominio, cioe' infinite nel caso di un dominio infinito.

Ad esempio, se consideriamo come f(x) la funzione identica, allora una restrizione finita g(x) potrebbe essere la funzione g(x)=x definita nel solo intervallo [0,10] ed indefinita altrove.

15.2 Terzo teorema di Rice

Come vedremo subito, il concetto di restrizione finita appare gia' nell'enunciato del terzo teorema di Rice. In questo paragrafo enunceremo e dimostreremo tale teorema.

3° teorema di Rice:

Sia $I \subseteq N$ un insieme che rispetta le funzioni. Se esiste una certa funzione calcolabile f tale che:

1. $\{x : \Phi_x = f\} \subseteq I$
2. $(\forall g)$, *se* $g \leq f$ *con g di dominio finito, allora* $\{x : \Phi_x = g\} \subseteq I$

allora I non e' semidecidibile.

Dim:

Come gia' per la dimostrazione del secondo teorema di Rice, il nostro obiettivo sara' dimostrare che !k e' riducibile a I. A questo scopo, consideriamo la seguente funzione in due variabili (dove f(y) e' la funzione calcolabile dell'ipotesi del teorema):

$$h(x,y) = \begin{cases} f(y) & \text{se } P_x \downarrow x \text{ in un num. di passi} \leq y \\ \uparrow & \text{altrimenti} \end{cases}$$

E' facile vedere che h(x,y) e' calcolabile. Infatti, per calcolarla, basta far partire il programma Px con input x e controllare cosa succede dopo y passi. Dopo tale periodo, se la computazione non e' terminata allora si fa partire il programma per f con input y, altrimenti si fa partire un programma che cicla all'infinito.

Trovando un programma per h(x,y), abbiamo quindi dimostrato la calcolabilita' di tale funzione. Possiamo quindi applicare ad essa il teorema del parametro. Esistera' quindi una funzione totale e calcolabile S(x) tale che $\Phi_{S(x)}(y)=h(x,y)$. Da cio' si ricava che:

- $x \in k \Rightarrow Px\downarrow x$ in C_0 passi $\Rightarrow S(x) \notin I$

  In questo caso avremo:

  $$\phi_{S(x)}(y)=\begin{cases} f(y) & \text{se } y \leq C_0 \\ \uparrow & \text{se } y > C_0 \end{cases}$$

  Questa funzione e' chiaramente una restrizione finita di f, quindi per l'ipotesi del teorema, S(x) non sta in I.

- $x \notin k \Rightarrow Px\uparrow x$ sempre $\Rightarrow \Phi_{S(x)}(y)=f(y) \Rightarrow S(x) \in I$.

  In questo caso, Px non termina su nessun input y, per cui, tanto piu', non termina in un numero di passi minore o uguale a y. Per questo motivo abbiamo che S(x) e' in pratica equivalente a f, per cui S(x) sta in I.

Abbiamo quindi dimostrato che S(x) e' la trasformazione che ci permette di risurre !k ad I, da cui deduciamo I e' non semidecidibile.

15.3 Decidibilita' dei programmi definiti su tutti gli input.

Consideriamo ora l'insieme A = {x : Φ_x e' totale} dei programmi definiti su tutti gli input. Chiaramente A rispetta le funzioni, in quanto una funzione equivalente ad una totale e' anch'essa totale. Inoltre, $A \neq \phi \neq N$, in quanto vi sono funzioni totali (ad esempio la funzione identica) ma non tutte le funzioni esistenti sono totali (ad esempio la funzione nulla non lo e').

Possiamo quindi applicare il primo teorema di Rice per concludere che **A non e' decidibile**. Inoltre, dal momento che la funzione nulla fa parte del complementare di A, possiamo anche concludere che **!A non e' semidecidibile**.

Il secondo teorema di Rice in questo caso non e' applicabile, perche' A e' composto da funzioni totali, e sappiamo che non esiste alcuna funzione che possa estendere una funzione totale (a parte se vogliamo la funzione stessa, che pero' certo non starebbe in !A).

Il terzo teorema di Rice e' invece applicabile se consideriamo la funzione identica come la nostra f. Infatti, tale funzione e' totale, mentre una restrizione finita della funzione identica non sara' certo totale, dunque stara' in !A. Da cio', deduciamo che **A non e' semidecidibile**, e quindi abbiamo finito lo studio di questo insieme.

15.4 Decidibilita' dei programmi che calcolano funzioni suriettive.

Consideriamo infine l'insieme I = {x : cod Φ_x = N} dei programmi che calcolano funzioni suriettive. Chiaramente I rispetta le funzioni, in quanto una funzione equivalente ad una suriettiva e' anch'essa suriettiva. Inoltre, $I \neq \phi \neq N$, in quanto vi sono funzioni suriettive (ad esempio la funzione identica) ma non tutte le funzioni esistenti sono suriettive (ad esempio la funzione nulla non lo e').

Possiamo quindi applicare il primo teorema di Rice per concludere che **I non e' decidibile**. Inoltre, dal momento che la funzione nulla fa parte del complementare di I (la funzione vuota non e' suriettiva), possiamo anche concludere che **!I non e' semidecidibile**.

Il secondo teorema di Rice in questo caso non e' applicabile, in quanto qualsiasi estensione di una funzione suriettiva sara' anch'essa suriettiva.

Il terzo teorema di Rice e' invece applicabile se consideriamo la funzione identica come la nostra f. Infatti, tale funzione e' suriettiva, per cui $f \subseteq I$, mentre una restrizione finita g della funzione identica, avendo dominio e codominio finiti, non sara' certo suriettiva, dunque $g \subseteq !I$. Da cio' deduciamo che **I non e' semidecidibile**, e quindi abbiamo finito lo studio di questo insieme.

16ª Lezione

In questa lezione ci occuperemo di chiarire il concetto di insieme ricorsivamente enumerabile.

16.1 Insiemi ricorsivamente enumerabili

Un insieme $A \subseteq N$ si dice **ricorsivamente enumerabile (r. e.)** se esiste una certa funzione f calcolabile totale tale che $A = cod(f)$.

Ad esempio, l'insieme dei numeri pari e' ricorsivamente enumerabile, in quanto li posso enumerare (con la sequenza 0, 2, 4, 6, 8, ...). Questo procedimento equivale a prendere la funzione calcolabile totale **f(x) =2x**, che ha come codominio proprio l'insieme dei numeri pari, per cui in questo modo viene soddisfatta la definizione di insieme r.e.

16.2 Equivalenza tra insiemi semidecidibili e insiemi r.e

In questo paragrafo vedremo che esiste un legame strettissimo tra insiemi semidecidibili ed insiemi r.e. Intuitivamente, dato un insieme A ricorsivamente enumerabile, posso semidecidere l'appartenenza di un certo x a tale insieme. Infatti, per farlo, basta enumerare tutti gli elementi di A fino a trovare x. Nel caso x non stesse in A si entrerebbe in un loop infinito.

Teorema:
Un insieme non vuoto $A \subseteq N$ e' r.e. se e solo se e' semidecidibile

Dim →:
Supponiamo di avere un insieme non vuoto A ricorsivamente enumerabile. Allora, per definizione, esiste una funzione calcolabile totale f, tale che $A = cod(f)$. Cio' detto, preso un qualsiasi x, possiamo calcolare f(0) grazie al programma che calcola f. Se $x = f(0)$ allora $x \in A$, altrimenti viene calcolato f(1). Se $x = f(1)$ allora $x \in A$, altrimenti viene calcolato f(2) e cosi' via. In questo modo, abbiamo scritto un programma per semidecidere A, in quanto, qualora x non stesse in A, si ciclerebbe all'infinito.

Dim ←:
Supponiamo di avere un insieme non vuoto A semidecidibile, allora la sua funzione semicaratteristica $X_A(x)$ e' calcolabile. Il nostro scopo, ora, e' di trovare una funzione calcolabile e totale f tale che $A = cod(f)$ partendo proprio da un programma per la funzione semicaratteristica. Se ci riusciremo avremo dimostrato che A e' ricorsivamente enumerabile.
Un algoritmo per la nostra funzione f, che vediamo visualizzato in figura 16.1, e' il seguente:
chiamiamo P un programma che semidecide A. Ora, spazziamo il piano seguendo il consueto procedimento. Iniziamo cioe' eseguendo P(0) per 10 minuti (la nostra unita' di tempo). Se in tale intervallo di tempo il programma termina, allora $0 \in A$, per cui pongo f(0)=0, il che' indica che la prima volta che il programma ha terminato l'ha fatto sull'input 0 (se l'avesse fatto sull'input 2 avremmo posto f(0)=2). In seguito, seguendo il nostro procedimento standard, eseguiamo P(1) per 10 minuti. Supponiamo che in questo caso il programma non termini. Allora eseguiamo P(0) per 20 minuti, il

quale, ovviamente termina (visto che ha terminato anche per 10). Cio' ci fa concludere che f(1)=0. Andiamo avanti cosi' e calcoliamo P(0) per 30 minuti, che termina, per cui f(2)=0. Supponiamo quindi, andando ancora avanti, che P(1) in 20 minuti termini, allora avremo f(3)=1 e cosi' via. In pratica il dominio di f e' l'insieme, ordinato, delle volte in cui P termina, e il codominio e' l'insieme di input su cui P e' terminato.

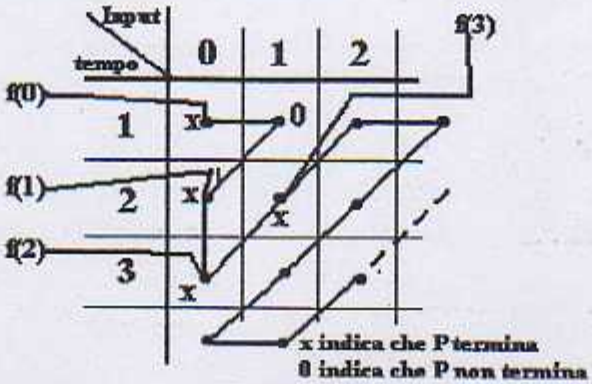

Figura 16.1

E' chiaro che, qualora P terminasse su un solo input k, la nostra f risulterebbe essere la funzione costante f(x) = k.

Vediamo adesso un altro utile teorema che associa il concetto di enumerazione a quello di decidibilita'. E' chiaro che un insieme finito e' facilmente enumerabile e, proprio per questo, decidibile, mentre e' meno evidente, a prima vista, il seguente enunciato:

Teorema:
Un insieme A infinito e' decidibile, se e solo se puo' essere enumerato in ordine strettamente crescente.

Dim ←:
Se posso enumerare un insieme A in modo strettamente crescente, allora, preso un qualsiasi x, potro' calcolare la sequenza $f(0) < f(1) < f(2) < \ldots < f(n), \ldots$ A questo punto, se vi e' un certo n tale che f(n)=x, allora $x \in A$. Altrimenti, si arriva ad un punto in cui $f(n-1) < x < f(n)$, nel qual caso concludiamo che $x \notin A$. Quindi abbiamo trovato un programma che decide A, per cui A e' decidibile.

Dim →:
Se A e' decidibile, allora prendo uno ad uno tutti I numeri naturali e li passo ad un programma P che decide A. Comincio quindi col calcolare P(0). Nel caso $0 \in A$ allora pongo f(0)=0. Quindi calcolo P(1). Supponiamo che il responso sia che $1 \notin A$. Allora calcolero' P(2). Se $2 \in A$ allora porro' f(1)=2 e cosi' via. Abbiamo cosi' ottenuto una funzione f calcolabile e totale strettamente crescente il cui codominio e' A' quindi abbiamo enumerato A in modo strettamente crescente.

17ª Lezione

Vediamo adesso un teorema che ci permettera' di definire una importante caratteristica degli insiemi semidecidibili. In seguito, parleremo di alcune impostanti proprieta' degli insiemi ricorsivamente enumerabili.

17.1 Una proprieta' degli insiemi semidecidibili

Teorema:
Dato un insieme I che rispetta le funzioni, con I≠ϕ ed I≠N:
Se I e' semidecidibile allora
$x \in I \Leftrightarrow (\exists g)$ *con dom(g) finito,* $g \leq \Phi_x$, *tale che* $\{y : \Phi_y = g\} \subseteq I$

Dim →:
Per ipotesi, abbiamo che I e' semidecidibile. Consideriamo un certo x∈I, tale che x sia il programma che implementa una certa funzione f. Quindi, dal momento che I rispetta le funzioni, avremo che $\{y : \Phi_y = f\} \subseteq I$.
Ora, supponiamo per assurdo che non esista alcuna funzione $g \leq \Phi_x$, con g di dominio finito, tale che $\{z : \Phi_z = g\} \subseteq I$.
Allora avremo che, per ogni g , $\{z : \Phi_z = g\} \subseteq !I$ (dal momento che I rispetta le funzioni, I programmi che calcolano g devono stare tutti in I o tutti in !I).
Ma quindi ci troveremmo di fronte ad una situazione in cui, data una funzione che sta in un insieme, tutte le sue restrizioni finite stanno nel suo complementare, per cui, per il terzo teorema di Rice, I risulterebbe non semidecidibile, il che' e' assurdo. Quindi esiste una restrizione finita g di f che verifica il teorema.

Dim ←:
Per ipotesi, abbiamo che I e' semidecidibile. Data una certa funzione f, supponiamo che esista un sua restrizione finita g tale che $\{y : \Phi_y = g\} \subseteq I$.
A questo punto, supponiamo per assurdo che un programma x che calcola f non stia in I (e quindi, dal momento che I rispetta le funzioni, nessun programma che calcola f stia in I).
Ma qui ci troveremmo di fronte ad una situazione in cui ci sarebbero due funzioni f e g con $g \leq f$, per le quali I programmi che calcolano g starebbero in I, mentre I programmi che calcolano f starebbero in !I, per cui, per il secondo teorema di Rice, potremmo dedurre che I non e' semidecidibile, il che e' assurdo. Quindi, dobbiamo per forza concludere che x∈I.

Chiariamo meglio il significato di questo teorema con un esempio pratico. Consideriamo l'insieme composto da tutte le funzioni che terminano su almeno un input, che come abbiamo visto nel paragrafo 11.1 e' semidecidibile. Allora, per il teorema appena dimostrato concludiamo che, presa una funzione f che termina su almeno un input, esiste una sua restrizione finita g che termina su almeno un input. Questo e' logico, in quanto, per ottenere una funzione g di questo tipo, basta prendere come dominio un valore x su cui la funzione f termina (e tale che g(x) = f(x)) e porre la funzione g ad indefinito su qualsiasi altro input.

In generale, un ragionamento di questo tipo si puo' effettuare per tutti gli insiemi semidecidibili. Ovvero, data una funzione f che soddisfa una certa proprieta', e' possibile trovare una sua restrizione finita g che soddisfi tale proprieta' semplicemente "tenendo" un numero finito di "input buoni" che soddisfano tale proprieta', e ponendo il resto ad indefinito.

Si puo' dimostrare, ma noi per semplicita' non lo facciamo, che non vale la parte inversa del teorema, cioe' se ci troviamo di fronte ad una funzione appartenente ad un insieme I tale che vi sia una sua restrizione finita appartenente ad I, non e' detto che I sia semidecidibile. Cio' si puo' dimostrare trovando un insieme I che risponda a tutti I requisiti del teorema ma che non sia semidecidibile (trovando, cioe', un controesempio), e cio', in generale, non e' semplice.

17.2 Proprieta' degli insiemi ricorsivamente enumerabili.

Vediamo adesso un teorema che ci permette di delineare alcune importanti caratteristiche degli insiemi ricorsivamente enumerabili.

Teorema:
Un insieme I e' ricorsivamente enumerabile se e solo se :
1. *I = cod(f), con f funzione calcolabile*
2. *I = dom(f), con f funzione calcolabile*
3. *I ={x: f(x) = c} per una certa funzione calcolabile f ed una costante c*
4. *Esiste un predicato R, decidibile, tale che $x \in I \Leftrightarrow (\exists y)$ tale che R(x,y) e' vero*

Dim (1,→):
Se I e' ricorsivamente enumerabile, allora per definizione esiste una funzione calcolabile totale f tale che I = cod (f).

Dim (1←):
Dato I = cod(f), con f funzione calcolabile, allora, il seguente algoritmo, ci permettere di semidecidere I:
consideriamo un certo programma P che calcola f. Dato un qualsiasi naturale x, cominciamo calcolando P sull'input zero per 10 minuti. Se P termina, allora confrontiamo il risultato della computazione con x. Nel caso I due valori siano uguali, concludiamo dicendo che $x \in I$, altrimenti continuiamo calcolando P sull'input 1 per 10 minuti, e cosi' via spazzando il piano input / tempo di computazione come mostrato nella figura 17.1. Nel caso x non stesse in I, allora ci troveremmo, giustamente, in un loop infinito.

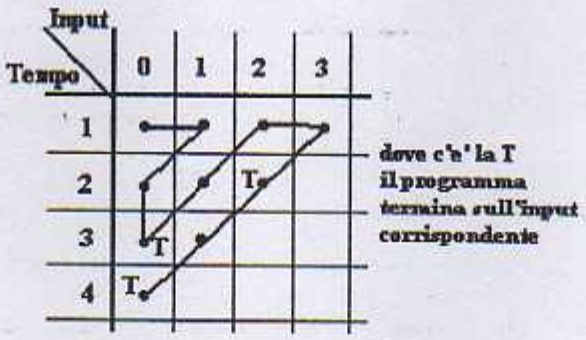

Figura 17.1

Quindi I risulta essere semidecidibile, e, quindi, ricorsivamente enumerabile.

Dim (2,←):
Se I e' il dominio di una funzione f calcolabile, allora I e' semidecidibile. Questo perche', se consideriamo un programma P che calcola f, e gli diamo in pasto un certo naturale x, esso terminera' se $x \in I$, mentre ciclera' all'infinito se $x \notin I$. Abbiamo cosi' trovato un algoritmo per semidecidere I, per cui I e' semidecidibile, quindi I e' anche r.e. per il teorema del paragrafo 16.2.

Dim (2,→):
Se un insieme I e' ricorsivamente enumerabile, allora, per il teorema del paragrafo 16.2, e' anche semidecidibile. Allora la sua funzione semicaratteristica $X_I(x)$ e' calcolabile, ed avra' per dominio proprio I.

Dim (3,→):
Se un insieme I e' ricorsivamente enumerabile, allora e' anche semidecidibile. Allora la sua funzione semicaratteristica $X_I(x)$ e' calcolabile, e, come sappiamo, tale funzione fornisce in output una costante (di solito 1) se $x \in I$.

Dim (3,←):
Consideriamo una funzione f calcolabile tale che I = {x : f(x) = c} e prendiamo un programma P che calcola f. Allora, I e' ricorsivamente enumerabile, in quanto I e' semidecidibile. Per vedere questo, basta che facciamo partire il

programma P su un qualsiasi input x. Se la computazione termina ed ha come risultato c, allora $x \in I$, altrimenti si entra nel solito ciclo infinito.

Dim (4,←):
Se I verifica la tesi di questo punto, allora I e' semidecidibile, e quindi r.e. , in quanto, se consideriamo un programma P che decide R allora, preso un qualsiasi naturale x, potremo dare in pasto a P, uno di seguito all'altro, gli input (x , 0) ; (x, 1) ; (x , 2) ; ... ; (x , i) ; ... Se una di queste coppie fara' terminare la computazione di P, allora tale coppia apparterra' ad R, e quindi $x \in I$.

Dim (4,→):
Se I e' r.e. , e quindi semidecidibile, esiste un programma P che semidecide I.
Ora, preso **$R = \{ (x,y) : P\downarrow x \text{ in} \leq y \text{ passi} \}$** , tale insieme e' decidibile, poiche', dato in pasto un qualsiasi input (x , y) a P, in al piu' y passi e' possibile sapere se $(x , y) \in R$ o meno. Cio' implica che x sta in I se e solo se esiste un tempo in cui la computazione termina (dal momento che P semidecide I, se $P\downarrow x$ allora $x \in I$).
Formalmente, avremo che, in questo caso, $x \in I \Leftrightarrow (\exists y)$ tale che R(x,y) e' vero, dove y rappresenta un tempo finito.

Da quest'ultimo punto del teorema possiamo trarre le seguenti considerazioni. In pratica, se consideriamo l'insieme decidibile R come mostrato in figura 17.2, I risulta essere la proiezione di tale insieme sull'asse X.

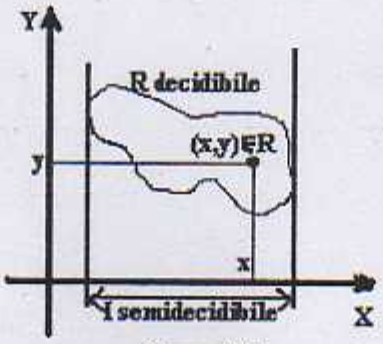

Figura 17.2

Infatti, preso un qualsiasi $x \in I$, deve esistere un y tale che $(x,y) \in R$, per cui x appartiene ad I se e solo se la retta perpendicolare ad X passante per (x,0) interseca, in almeno un punto, la superficie R. Ovviamente questo ragionamento puo' essere fatto anche invertendo l'ordine degli assi. In parole povere possiamo affermare che **un insieme semidecidibile e' la proiezione di un insieme decidibile.**