18ª Lezione

In questo capitolo vedremo di enunciare e spiegare uno dei teoremi piu' importanti della teoria della calcolabilita', ovvero il **secondo teorema di ricorsione**, che, come il primo teorema di ricorsione che vedremo nel prossimo capitolo, e' dovuto a Kleene.

18.1 Enunciato e dimostrazione del secondo teorema di ricorsione

Cominciamo con l'enunciare e dimostrare il secondo teorema di ricorsione, cercando di chiarire il significato di tale teorema. Nei prossimi paragrafi vedremo poi delle importanti conseguenze di tale teorema.

In pratica, grazie a questo teorema Kleene ha dimostrato che, presa una qualunque funzione calcolabile totale h che trasforma programmi in programmi (ovvero del tipo N→N) esiste sempre un programma che viene trasformato da tale funzione in uno equivalente, a prescindere dal tipo di codifica che adottiamo per associare I programmi ai numeri naturali.

2° teorema di ricorsione:
Data una qualsiasi funzione calcolabile totale h, allora esiste un certo n per cui:
$\Phi_n = \Phi_{h(n)}$

Dim:
Data una qualsiasi funzione calcolabile e totale h, consideriamo la seguente funzione in due variabili f(x,y):

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

Essendo h totale, questa funzione e' chiaramente calcolabile, in quanto e' il risultato di applicazioni di funzioni calcolabili a numeri naturali (Al massimo potremmo avere che Φ(x,x) cicli indefinitivamente per qualche x, ma cio' non ci interessa, anche perche' sappiamo che esistono funzioni che terminano su se stesse, per cui il dominio di f(x,y) e' senz'altro un insieme non vuoto. Questo perche', essendo h una funzione totale, sicuramente tutti gli output che escono da Φ(x,x) vengono "catturati" da h in input e generano un risultato). Possiamo quindi applicare ad f(x,y) il teorema del parametro. Esiste quindi una certa funzione calcolabile totale S(x) tale che:

$$\Phi_{S(x)}(y) = f(x,y) = \Phi_{h(\Phi x(x))}(y) \qquad (1)$$

Ora, se consideriamo un programma m che calcola S(x), avremo che $S(m) = \Phi_m(m)$, per cui, se chiamiamo m la quantita' $\Phi_m(m)$, avremo, sostituendo m al posto di x nella (1), che:

$$\Phi_{\Phi m(m)}(y) = f(x,y) = \Phi_{h(\Phi m(m))}(y) \quad \Rightarrow \quad \Phi_n(y) = \Phi_{h(n)}(y)$$

Abbiamo dunque ricavato il risultato che volevamo, dimostrando cosi' il secondo teorema di ricorsione.

Nei prossimi paragrafi, ci preoccuperemo di studiare alcune conseguenze di questo teorema e di trattare qualche esempio di applicazione dello stesso.

18.2 Dimostrazione del fatto che k non rispetta le funzioni

Consideriamo la seguente funzione in due variabili f (x,y):

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

E' chiaro che questa funzione e' calcolabile, in quanto e' facile decidere se x=y. Possiamo quindi applicare ad essa il teorema del parametro. Esiste quindi una funzione calcolabile e totale S(x) tale che $\Phi_{S(x)}(y)=f(x,y)$. Quindi, per il secondo teorema di ricorsione (in cui la funzione h calcolabile e totale che consideriamo e' S(x)) esiste un certo numero naturale n tale che $\Phi_{S(n)}(y) = \Phi_n(y) = f(n,y)$. Ma, visto come e' fatta f(x,y), f(n,y) generera' output solo se n=y. Cio' comporta che $\Phi_n(y)$ converge solo su se stessa, per cui $\Phi_n \in k$.

Ora, qualsiasi programma Φ_k equivalente a Φ_n che possiamo prendere, converge sempre e solo su n, non su k (su se stesso), per cui $\Phi_k \notin k$. Quindi, un elemento di k puo' essere equivalente ad un elemento di !k. Cio' comporta che k non rispetta le funzioni.

18.3 Dimostrazione di Rice 1 tramite il secondo teorema di ricorsione

Consideriamo un insieme I che rispetta le funzioni, tale che I sia diverso da N e dall'insieme vuoto. Consideriamo, inoltre, la seguente funzione g(x):

$$g(x)=\begin{cases}C0 & \text{se } x\in I \quad \text{con} \quad C0\in\overline{I}\\ C1 & \text{se } x\notin I \quad \text{con} \quad C1\in I\end{cases}$$

Questa funzione ha ovviamente senso solo se I e' un insieme decidibile. Facciamo finta, per assurdo che lo sia. Allora avremo che g(x) e' una funzione calcolabile totale, per cui possiamo applicare ad essa il secondo teorema di ricorsione. Esistera' quindi un certo numero naturale n tale che $\Phi_n = \Phi_{g(n)}$. Ora, si potranno avere due casi:

- Se $n\in I \Rightarrow \Phi_n = \Phi_{g(n)} = \Phi_{C0}$,ma C0 sta in !I, e cio' e' assurdo perche' I rispetta le funzioni
- Se $n\notin I \Rightarrow \Phi_n = \Phi_{g(n)} = \Phi_{C1}$,ma C1 sta in I, e cio' e' assurdo perche' I rispetta le funzioni

Questa dimostrazione per assurdo prova che I non e' decidibile, quindi abbiamo dimostrato parte del primo teorema di Rice tramite il secondo teorema di ricorsione.

18.4 Costruzione di un programma che genera se stesso

Consideriamo la funzione in due variabili f(x,y)=x. Questa funzione e' chiaramente calcolabile, per cui posso applicare ad essa il teorema dl parametro. Esiste quindi una funzione calcolabile e totale S(x) tale che $\Phi_{S(x)}(y)=f(x,y)$. Ora, dal momento che S(x) e' calcolabile e totale, soddisfa alle condizioni del secondo teorema di ricorsione, per cui esiste un certo numero naturale n tale che:

$$\Phi_{S(n)}(y) = \Phi_n(y) = f(n,y) = n$$

Quindi, $\Phi_n(y)$ e' un programma che, qualsiasi input gli venga fornito, resituisce come risultato sempre se stesso.

18.5 Dimostrazione del fatto che k non e' decidibile

Supponiamo per assurdo che k sia decidibile. Consideriamo ora la seguente funzione:

$$g(x)=\begin{cases}a \text{ se } x\in k & \text{con a programma per la funzione vuota}\\ b \text{ se } x\notin k & \text{con b programma per la funzione identica}\end{cases}$$

Ovviamente, se consideriamo k decidibile, g(x) risulta essere una funzione calcolabile e totale. Cio' ci permette di applicare a g(x) il secondo teorema di ricorsione. Esiste quindi un certo numero naturale n tale che $\Phi_{g(n)} = \Phi_n$. A questo punto, ci troviamo di fronte a due possibilita':

- Se $n \in k$ allora $g(n)=a \Rightarrow \Phi_n = \Phi_{g(n)} = \Phi_a$ che e' assurdo perche' Φ_n termina su se stesso mentre Φ_a, calcolando la funzione vuota, non termina su se stesso.
- Se $n \notin k$ allora $g(n)=b \Rightarrow \Phi_n = \Phi_{g(n)} = \Phi_b$ che e' assurdo perche' Φ_n non termina su se stesso mentre Φ_b, calcolando la funzione identica, termina su se stesso.

Quindi considerare k decidibile ci ha portato a delle contraddizioni, il che prova che **k non e' decidibile.**

18.6 Dimostrazione che esistono infiniti naturali n tali che $\Phi_n = \Phi_{h(n)}$

Col secondo teorema di ricorsione abbiamo visto che per ogni funzione h calcolabile e totale si ha che esiste un n per cui $\Phi_n = \Phi_{h(n)}$. Dimostriamo ora che in realta' questi n sono infiniti. Per far questo, consideriamo la seguente funzione:

$$h'(x) = \begin{cases} CD & \text{se } 0 \leq x \leq n \quad \text{Con CD non equivalente ai programmi codificati da } 0,1,2,\ldots,n \\ h(x) & \text{se } x > n \end{cases}$$

Ora, questa funzione e' calcolabile e totale, quindi soddisfa alle condizioni del secondo teorema di ricorsione. Cio' significa, in generale, che esiste un certo numero naturale n', che per il momento non sappiamo se diverso da n, tale che $\Phi_{h'(n')} = \Phi_{n'}$. A questo punto, ci troviamo di fronte a due possibilita':

- Se $0 \leq n' \leq n$ allora $\Phi_{n'} = \Phi_{h'(n')} = \Phi_{CD}$ ma questo e' assurdo perche' $\Phi_{n'}$ e' un programma compreso tra 0 ed n, mentre, Φ_{CD} e' un programma non equivalente ad alcuno di detti programmi, quindi neanche a $\Phi_{n'}$. **Quindi n' > n.**
- Se $n' > n$ allora $\Phi_{n'} = \Phi_{h'(n')} = \Phi_{h(n')}$ Quindi questa e' l'unica situazione che puo' verificarsi. Cio' significa che n' e' un nuovo numero naturale, diverso da n, tale che $\Phi_{h(n')} = \Phi_{n'}$.

Abbiamo quindi dimostrato che, partendo da un n che soddisfa la tesi del secondo teorema di ricorsione, possiamo sempre trovare un nuovo naturale n', diverso da n, con la stessa proprieta'. E' chiaro che questo ragionamento puo' essere ripetuto un numero indefinito di volte. Ci sono quindi **infiniti** numeri naturali che soddisfano la tesi del secondo teorema di ricorsione.

19ª Lezione

Nel capitolo 18 abbiamo studiato il secondo teorema di ricorsione, mentre nel capitolo 20 studieremo il primo teorema di ricorsione. In questo capitolo, cercheremo invece di capire quale meccanismo colleghi questi due teoremi, dovuti a Kleene, al concetto di ricorsione che abbiamo gia' visto quando abbiamo parlato di ricorsione primitiva nei capitoli 4 e 5.

19.1 Nuovi tipi di definizioni di funzioni per ricorsione

Abbiamo visto la definizione di una funzione per ricorsione primitiva. Diremo che una certa funzione f(x,y) e' definita per **ricorsione primitiva** se esistono due funzioni calcolabili g(x) ed h(x) tali che:

$$f(x,y) = \begin{cases} g(x) & \text{se } y = 0 \\ h(x,y, f(x,y-1)) & \text{se } y > 0 \end{cases}$$

Vediamo adesso un altro esempio di definizione per ricorsione (in questo caso riconducibile alla ricorsione primitiva), di una funzione ad una variabile:

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

Queste due definizioni di ricorsione si basano entrambe sul principio di induzione, in quanto partendo dal valore di una certa f(x) si arriva al valore di f(x+1). Vediamo adesso un esempio di definizione ricorsiva che non si basa sul principio di induzione. Consideriamo la seguente definizione su una funzione in due variabili:

$$f(x,y) = \begin{cases} 1 & \text{se } x = 0 \\ f(x-1, f(x,y)) & \text{se } x > 0 \end{cases} \qquad (1)$$

In questo caso, la definizione ricorsiva puo' essere interpretata in due modi diversi. Vediamo un pratico esempio numerico, in cui cerchiamo di calcolare il valore di f(1,0):

- $f(1,0) = f(0,f(1,0)) = 1$ — Se consideriamo il simbolo f **piu' esterno**
- $f(1,0) = f(0,f(0,f(0,f, \ ) \ldots)) = \uparrow$ — Se consideriamo il simbolo f **piu' interno** e lo sviluppiamo ogni volta. In questo caso questa operazione porta ad un ciclo infinito.

Quindi, nel caso di definizioni ricorsive piu' complesse della classica ricorsione primitiva, ci si deve porre anche il problema di sviluppare prima I simboli di funzione piu' interni piuttosto che quelli piu' esterni o viceversa. La questione non e' banale in quanto, come si e' appena visto, questo puo' comportare la terminazione o meno del programma. Queste strategie di sviluppo dei simboli sono dette **strategie di riduzione** e sono di due tipi:

- **Outermost:** si sceglie il simbolo di funzione **piu' esterno** da sviluppare
- **Innermost:** si sceglie il simbolo di funzione **piu' interno** da sviluppare

Si puo' dimostrare, ma in realta' e' facile convincersene, che nel caso di una definizione per ricorsione primitiva, non vi sono scelte di simboli di funzione piu' interni o piu' esterni da sviluppare, perche', ogni volta, ve ne e' solo uno. Infatti, in una ricorsione primitiva, f(x) sara' sempre definito in termini di f(x-1), mai in termini di f(x).

19.2 Parallelismo tra ricorsione primitiva e secondo teorema di ricorsione

Consideriamo la seguente funzione in una variabile definita per ricorsione primitiva:

$$g(x)=\begin{cases}0 & \text{se } x=0\\ g(x-1)+3 & \text{se } x>0\end{cases}$$

A questo punto, il nostro scopo e' quello di trovare una funzione **g(x):N→N calcolabile** che soddisfi la definizione ricorsiva di cui sopra. Cio' vuol dire che la funzione g(x) che troveremo, sostituita nella parte destra della definizione ricorsiva, dovra' fornire come risultato a sinistra g(x) stessa. Per ricavare g(x) procediamo, inizialmente, per tentativi:

h(x) = funzione identica

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

h(x) = funzione successore

$$g(x)=x+1\neq\begin{cases}0 & \text{se } x=0\\ x+3 & \text{se } x>0\end{cases}$$

Dunque la funzione identica e la funzione successore non sono cio' che cerchiamo, in quanto sostituite alla definizione ricorsiva non generano l'identita'. Vediamo adesso invece una funzione che ci dara' maggiori soddisfazioni:

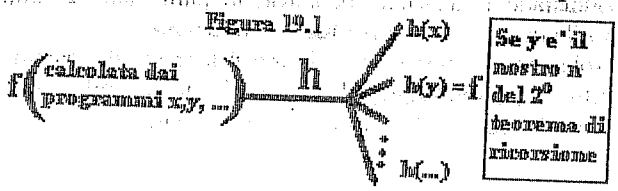
h(x) = funzione che triplica

$$g(x)=\begin{cases}0 & \text{se } x=0\\ 3x & \text{se } x>0\end{cases}=\begin{cases}0 & \text{se } x=0\\ 3(x-1)+3=3x & \text{se } x>0\end{cases}$$

Quindi la funzione che triplica e' quella che cerchiamo, in quanto sostituita a sinistra e a destra della definizione ricorsiva fornisce l'identita'. Cio' si riassume dicendo che **g(x) e' la soluzione dell'equazione ricorsiva**. Parallelamente, se chiamiamo **h** la codifica del programma che permette di calcolare g(x) ricorsivamente (calcolando g(0), g(1), g(2), ... , fino a g(x), diremo che **h e' un metodo di trasormazione** da funzioni calcolabili a funzioni calcolabili che permette di trasformare una certa funzione calcolabile in se stessa.

Se osserviamo bene questa definizione di g(x) ed h, ci accorgiamo che essa ha molte affinita' con le definizioni di n ed h date nel secondo teorema di ricorsione. Infatti, anche in questo caso h puo' essere vista come una trasformazione da funzioni calcolabili a funzioni calcolabili, mentre n, essendo trasformato da h in se stesso, puo' essere tranquillamente messo in analogia con g(x) (si noti che, **se consideriamo h come una trasformazione da funzioni a funzioni, essa non e' piu' una funzione**, in quanto, presa una certa funzione f in input, essa produce in output l'insieme di tutti I programmi che rappresentano l'applicazione della **funzione** h ai programmi che implementano f, come si vede in figura 19.1). Quindi essa fa associare piu' output ad un solo input.

Figura 19.1



Forse la maggiore differenza che si nota nei due approcci al problema, e' che nel caso dell'approccio ricorsivo la soluzione dell'equazione ricorsiva e' una funzione (f nell'esempio di figura 19.1) mentre quando applichiamo il 2° teorema di ricorsione la soluzione della nostra equazione e' un programma (y nel caso di figura 19.1).

Abbiamo quindi visto l'analogia che passa tra gli oggetti g(x), h di una definizione per ricorsione primitiva, e gli oggetti n, h derivanti dall'applicazione del secondo teorema di ricorsione. Cio' ci suggerisce che, dal momento che siamo riusciti a trovare una soluzione per un'equazione ricorsiva tramite applicazione della ricorsione primitiva, dobbiamo poterci riuscire anche tramite secondo teorema di ricorsione. E' chiaro che, mentre nel primo caso trovare una soluzione significa trovare una funzione (g(x) = 3x nel nostro esempio), nel secondo ci aspettiamo di trovare come soluzione un programma (uno dei tanti che implementa g(x)). A questo scopo, chiamiamo y un generico programma che implementa g(x). Avremo:

$$\phi_y(x) = \begin{cases} 0 & \text{se } x = 0 \\ \phi_y(x-1)+3 & \text{se } x > 0 \end{cases}$$

Chiamiamo $\tau(x,y)$ la seguente funzione calcolabile:

$$\tau(x,y) = \phi_y(x) = \begin{cases} 0 & \text{se } x = 0 \\ \phi_y(x-1)+3 & \text{se } x > 0 \end{cases}$$

Questa funzione e' chiaramente calcolabile. Possiamo quindi applicare ad essa il teorema del parametro. Esiste quindi, parametrizzando rispetto a y, una funzione S(y) calcolabile e totale tale che $\Phi_{S(y)}(x) = \tau(x,y)$. Ora, possiamo considerare S come un metodo di trasformazione da funzioni calcolabili a funzioni calcolabili. In quest'ottica, se riusciamo a trovare un certo n per cui n=S(n), allora n sara' una soluzione per la nostra equazione ricorsiva. Ma, dal momento che S e' calcolabile e totale, per il secondo teorema di ricorsione abbiamo proprio che esiste un n tale che:

$$\phi_n = \phi_{S(n)}(x) = \begin{cases} 0 & \text{se } x = 0 \\ \phi_n(x-1)+3 & \text{se } x > 0 \end{cases}$$

Per cui Φ_n e' una funzione che e' soluzione della nostra definizione ricorsiva (Φ_n=g(x)=3x nel nostro esempio), mentre n e' il programma che implementa tale funzione e che risponde a tutti I requisiti del secondo teorema di ricorsione. Purtroppo, mentre Φ_n in qualche modo e' ricavabile (ad esempio per tentativi come abbiamo fatto noi), n e' un programma di cui conosciamo solo l'esistenza, ma di cui non sappiamo nient'altro.

19.3 Problemi nel calcolo delle soluzioni di equazioni ricorsive

Un problema nel calcolo delle soluzioni di equazioni ricorsive lo abbiamo appena visto, e consiste nel fatto che applicando il secondo teorema di ricorsione spesso di queste soluzioni possiamo solo assicurare l'esistenza.

Un altro problema lo si ha con definizioni ricorsive del tipo (1) viste nel paragrafo 19.1, in cui, a seconda della strategia di riduzione adottato, possiamo ottenere soluzioni diverse, ovvero diverse funzioni calcolabili che la soddisfano. Ad esempio, se nell'esempio (1) del paragrafo 19.1 vogliamo calcolare f(1,y) avremo:

Strategia OUTERMOST

$$f_1(1,y) = \begin{cases} 1 & \text{se } x = 0 \\ 1 & \text{se } x > 0 \end{cases}$$

Strategia INTERMOST

$$f_2(1,y) = \begin{cases} 1 & \text{se } x = 0 \\ \uparrow & \text{se } x > 0 \end{cases}$$

Entrambe queste soluzioni soddisfano l'equazione ricorsiva, in particolare, **la soluzione ottenuta con la strategia Intermost e' la meno definita** (la piu' "piccola") **tra tutte le soluzioni.**

Ovviamente, per trovare una soluzione, e' possibile utilizzare il teorema del parametro e quindi applicare il secondo teorema di ricorsione, come fatto nel paragrafo precedente, ma cio' in generale sara' utile solo per dimostrare l'esistenza di una soluzione, e non per trovarla, come del resto abbiamo gia' detto.

20ª Lezione

In questo capitolo cercheremo di fissare alcuni concetti molto importanti per il calcolo delle soluzioni di equazioni ricorsive. Cominciamo con la definizione di funzione estensionale e di funzionale.

20.1 Funzioni estensionali

Una funzione h : N→N totale si dice **estensionale** se $\Phi_x = \Phi_y \Rightarrow \Phi_{h(x)} = \Phi_{h(y)}$. Quindi una funzione estensionale trasforma programmi equivalenti in programmi equivalenti.

Se vediamo una funzione h estensionale sotto l'ottica di un metodo di trasformazione da funzioni calcolabili a funzioni calcolabili, essa e' una trasformazione molto diversa dalle altre, in quanto traforma una funzione in un insieme di programmi equivalenti, e non in un insieme di programmi disomogenei come nell'esempio visto in figura 19.1. Formalmente, data una funzione totale h (non necessariamente estensionale), se consideriamo l'applicazione della **trasformazione** h ad una funzione f, scriveremo:

$$f \xrightarrow{\tau_h} \tau_h(f)$$

Questa scrittura significa, come nell'esempio di figura 19.1, che una certa funzione f, implementata dai programmi x, y, ... , viene trasformata, dalla trasformazione h, nell'insieme di programmi h(x), h(y), ... , con h(x), h(y),..., equivalenti nel caso h sia estensionale. D'ora in poi, per non fare confusione tra la funzione h e la trasformazione h, useremo in quest' ultimo caso il simbolo τ_h (come si vede nella figura qui sopra) detto **funzionale**, che ha proprio lo scopo di indicare questo concetto. E' chiaro comunque che, allo scopo di trovare le soluzioni di equazioni ricorsive, I funzionali su cui porremo particolare attenzione saranno quelli associati a funzioni estensionali.

Esempio 1:
Consideriamo la funzione identita' **Id(x)=x**.
Questa funzione e' calcolabile e totale. Inoltre, abbiamo che $\Phi_x = \Phi_y \Rightarrow \Phi_{Id(x)} = \Phi_x = \Phi_y = \Phi_{Id(y)}$, quindi Id e' una funzione estensionale. Non solo, ma se consideriamo il **funzionale identico** τ_{Id} ad essa associato avremo:

$$f \xrightarrow{\tau_{Id}} \tau_{Id}(f) = f$$

Come vedremo piu' avanti, trovare la soluzione di un'equazione ricorsiva associata ad un certo funzionale τ_S , significa trovare una certa funzione f tale che τ_S (f) = f. Nel caso della funzione identica, qualunque funzione f prendiamo questa condizione viene sempre soddisfatta, quindi qualunque f e' soluzione dell'equazione ricorsiva identica f(x) = f(x).

Esempio 2:
Consideriamo la funzione costante **C0(x) = c0**.

Questa funzione e' calcolabile e totale. Inoltre, abbiamo che $\Phi_x = \Phi_y \Rightarrow \Phi_{C0(x)} = \Phi_{c0} = \Phi_{C0(y)}$, quindi C0 e' una funzione estensionale.

20.2 Come provare l'esistenza di una soluzione di un'equazione ricorsiva

Riassumiamo l'algoritmo che abbiamo esposto nel paragrafo 19.2 per trovare una soluzione di un'equazione ricorsiva. Esso si basa sul concetto di sostituire la funzione calcolabile che cerchiamo con un programma che la calcola, il quale ovviamente, essendo un numero, e' piu' facile da manipolare. Ad esempio, consideriamo la funzione:

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

Ora, come detto, sostituiamo ad f(x,y) un programma z che la calcola. Avremo:

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

Questa nuova funzione calcolabile puo' anche essere vista come una funzione in 3 variabili, e cioe':

$$\tau(x,y,z) = \begin{cases} 1 & \text{se } x = 0 \\ \phi_z((x-1), \phi_z(x,y)) & \text{se } x > 0 \end{cases}$$

Essendo questa funzione calcolabile, posso applicare ad essa il teorema del parametro, parametrizzando rispetto al programma corrispondente alla funzione che cerchiamo, cioe' z. Esiste quindi una funzione S(z) calcolabile e totale tale che $\Phi_{S(z)}(x,y) = \tau(x,y,z)$. Ora, e' chiaro che se troviamo un certo numero naturale z tale che $\Phi_z(x,y) = \Phi_{S(z)}(x,y)$ allora Φ_z e' la soluzione dell'equazione ricorsiva che cerchiamo. L'esistenza di tale z e' assicurata dal secondo teorema di ricorsione, mentre non sappiamo al momento nulla su chi sia Φ_z .

20.3 Qualcosa in piu' sulle funzioni estensionali

Osservando bene la funzione $\tau(x,y,z)$, notiamo che la funzione S(z) del teorema del parametro e' estensionale, in quanto $\Phi_z = \Phi_{z'} \Rightarrow \Phi_{S(z)} = \Phi_{S(z')}$ (per convincersene basta sostituire z con z' nella definizione di $\tau(x,y,z)$, allora si scopre che $\tau(x,y,z) = \tau(x,y,z')$, il che' porta alla tesi).

Abbiamo trovato quindi una situazione in cui la funzione S del teorema del parametro e' estensionale. Questo, in generale, non e' sempre vero, ad esempio, se consideriamo:

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

qui S(x) non e' estensionale, in quanto si ha che $\Phi_x = \Phi_{x'} \Rightarrow \Phi_{S(x)} = x \neq x' = \Phi_{S(x')}$, mentre se consideriamo:

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

allora S(x) e' estensionale, in quanto si ha che $\Phi_x = \Phi_{x'} \Rightarrow \Phi_{S(x)} = \Phi_y = \Phi_{S(x')}$.

Da questi esempi possiamo dedurre che, qualunque sia la funzione f(x,y) cui si applica il teorema del parametro, **S e' estensionale se e solo se x appare al massimo come indice** sia a sinistra che a destra della definizione ricorsiva. Ad esempio, la seguente funzione e' estensionale:

$$\phi_{S(x)}(y) = f(x,y) = \begin{cases} 1 & \text{se } x = 0 \\ \phi_x(y, \phi_x(y-1)) & \text{se } x > 0 \end{cases}$$

perche', sia a destra che a sinistra, gli x sono sempre indici e mai argomenti.

21ª Lezione

In questo capitolo termineremo lo studio delle funzioni ricorsive introducendo, tra l'altro, il primo teorema di ricorsione.

Cominciamo dando un paio di definizioni:

- Un **punto fisso** di un funzionale e' una qualsiasi funzione g tale che $\tau_h(g) = g$. Ricordandoci anche cio' che avevamo detto lo scorso capitolo, ogni soluzione di un'equazione ricorsiva, anche la funzione fix_τ di cui parleremo nel primo teorema di ricorsione, e' un punto fisso del funzionale associato all'equazione stessa.
- Se una funzione h e' calcolabile, totale, estensionale allora il funzionale τ_h e' detto **funzionale ricorsivo**, e gode della proprieta' $\tau_h(\Phi_x) = \Phi_{h(x)}$.

21.1 Il 1° teorema di ricorsione

Prima di enunciare e dimostrare il primo teorema di ricorsione, dimostriamo il seguente lemma:

Lemma 1:

Sia h una funzione calcolabile, totale, estensionale, allora $\tau_h(f)(x) = y$ se e solo se $(\exists \vartheta)$, con ϑ restrizione finita di f, tale che $\tau_h(\vartheta)(x) = y$

Dim:

Poiche' h e' estensionale, allora τ_h e' un funzionale ricorsivo, dunque se $f = \Phi_z$ allora $\tau_h(f) = \Phi_{h(z)}$. Quindi, intuitivamente, se per un certo x abbiamo che $\Phi_{h(z)}(x) = y$, concludiamo che se carichiamo in memoria solo gli output di f che fanno si che x vi sia definita, il risultato della computazione sara' sempre y.

Formalmente, consideriamo l'insieme $A = \{z : \tau_h(\Phi_z)(x) = y\}$. Questo insieme rispetta le funzioni, infatti:

$\Phi_z = \Phi_{z'}$ e $z \in A \Rightarrow z' \in A$ poiche' $\tau_h(\Phi_z) = \tau_h(\Phi_{z'})$.

Ora, $A \neq \phi$ perche' tutti I programmi che calcolano f stanno in A. In generale, pero', non sappiamo se $A \neq N$, per cui non possiamo applicare ad A il primo teorema di Rice. Quindi, per vedere se A e' semidecidibile, dobbiamo scrivere un algoritmo che semidecide A. Tale algoritmo e' il seguente:

Preso un qualsiasi input z, lo diamo in pasto al programma che calcola h ed otteniamo h(z). Quindi codifichiamo h(z) nel programma relativo. Infine, spazziamo il piano input-tempo lanciando quest'ultimo programma su tutti gli input. Se troviamo un input x per cui il programma termina e da' come risultato y, allora restituiamo 1, altrimenti continuiamo a ciclare. Questo programma, ovviamente, semidecide A.

Nel paragrafo 17.1 abbiamo visto che, dato A semidec., $z \in A \Leftrightarrow (\exists \vartheta$, con ϑ restriz. finita di f, tale che $\{y: \Phi_y = \vartheta\} \subseteq A)$.

Leggendo in maniera diversa questo teorema, possiamo dire che:

se $z \in A$, cioe' $\tau_h(\Phi_z)(x) = y$, allora z e' un programma per f, per cui $\tau_h(f)(x) = y$. Ma, per il teorema appena visto, sia ha che **questo e' vero se e solo se** esiste una funzione ϑ, restrizione finita di f, tale che $\{y : \Phi_y = \vartheta\} \subseteq A$, il che' equivale a dire che $\tau_h(\vartheta)(x) = y$. Questo conclude la dimostrazione.

1° teorema di ricorsione:
Sia h una funzione calcolabile, totale ed estensionale (quindi h definisce una trasformazione di funzioni in funzioni). Sia poi τ_h il funzionale che effettua la trasformazione $f \rightarrow \tau_h(f)$. Allora esiste una funzione calcolabile $fix_\tau : N \rightarrow N$ tale che:

1. *$\tau_h(fix_\tau) = fix_\tau$, cioe' $\Phi_z = fix_\tau \Rightarrow \Phi_z = \Phi_{h(z)}$*
2. *Se $\tau_h(g) = g$ allora $fix_\tau \leq g$, cioe' fix_τ e' la piu' piccola soluzione dell'equazione ricorsiva*
3. *Esiste un metodo effettivo per calcolare fix_τ.*

Dim (3):
Per prima cosa vediamo un modo per ricavare fix_τ. Piu' avanti dimostreremo che fix_τ e' calcolabile.
Costruiamo una successione di funzioni $f_0, f_1, \ldots, f_n, \ldots$ tale che:

$f_0 = f_\phi$
$f_1 = \tau_h(f_0)$
$f_2 = \tau_h(f_1)$
|
|
$f_n = \tau_h(f_{n-1})$

Dimostriamo che questa e' una successione di funzioni tale che f_i estende f_{i+1}, cioe' $f_0 \leq f_1 \leq \ldots \leq f_n$. Procediamo per induzione:

- **Caso base:** E' chiaro che $f_0 \leq f_1$, in quanto f_0 calcola la funzione vuota, che e' estesa da qualsiasi funzione. Supponiamo che vi sia un x per cui $f_1(x)=y$, da cui avremo che $\tau_h(f_0)(x) = f_1(x)=y$. Allora, per via del lemma 1, esiste una restrizione finita di f_0, che chiameremo ϑ, tale che $\tau_h(\vartheta)(x) = y$. Ma allora, dal momento che f_0 e' la funzione vuota, anche ϑ e' la funzione vuota, per cui ϑ e' una restrizione finita di f_1. Per cui, dal momento che esiste una restrizione finita ϑ di f_1 tale che $\tau_h(\vartheta)(x)=y$ allora, ripercorrendo al contrario il lemma 1, ricaviamo che $\tau_h(f_1)(x)=y$, cioe' $f_2(x)=y$, per cui $f_1 \leq f_2$.

- **Induzione:** Supponiamo per ipotesi induttiva che $f_0 \leq f_1 \leq \ldots \leq f_n$ e cerchiamo di dimostrare che $f_n \leq f_{n+1}$. Poniamo che vi sia un x per cui $f_n(x)=y$, da cui avremo che $\tau_h(f_{n-1})(x) = f_n(x)=y$. Allora, per via del lemma 1, esiste una restrizione finita di f_{n-1}, che chiameremo ϑ, tale che $\tau_h(\vartheta)(x) = y$. Ma allora, poiche' $\vartheta \leq f_{n-1}$ e $f_{n-1} \leq f_n$ per ipotesi induttiva, allora ϑ e' una restrizione finita di f_n. Per cui, dal momento che esiste una restrizione finita ϑ di f_n tale che $\tau_h(\vartheta)(x)=y$ allora, ripercorrendo al contrario il lemma 1, ricaviamo che $\tau_h(f_n)(x)=y$, cioe' $f_{n+1}(x)=y$, per cui $f_n \leq f_{n+1}$.

A questo punto, definiamo la nostra funzione fix_τ nel seguente modo:

$$fix_\tau = \bigcup_{i=0}^{+\infty} fi$$

Dimostriamo ora che la funzione fix_τ cosi' costruita e' calcolabile. Allo scopo, formiamo un algoritmo che la calcola:
Sia e(0) un programma che calcola f_0, cioe' la funzione vuota.
Definiamo ora per ricorsione primitiva una nuova funzione che, preso in input un cero n, dia in uscita **e(n) = un programma per f_n**:

$$e(n) = \begin{cases} e(0) = \lceil f_0 \rceil & n = 0 \\ e(n+1) = h(e(n)) & n > 0 \end{cases}$$

La funzione e(n), quindi, genera per ricorsione primitiva tutti I programmi per le f_n.
Ora, per come e' fatta fix_τ, avremo che $fix_\tau(x) = y$ se e solo se esiste un n per cui $\Phi_{e(n)}(x)=y$. Quindi, quello che vogliamo fare, e' calcolare $\Phi_{e(n)}(x)$ per ogni n, spazzando il piano **Programmi e(n) - Input x**. Se, una volta avviato questo procedimento, un certo programma e(k) termina su un certo input x, allora il risultato y della computazione e' il valore cercato per $fix_\tau(x)$, altrimenti si continua a ciclare all'infinito, il che' sarebbe giusto perche' significherebbe che fix_τ non e' definito su x.
Quindi fix_τ e' una funzione calcolabile.

Dim (1):

Dimostriamo ora che $\tau_h(\text{fix}_\tau) = \text{fix}_\tau$, cioe' che fix_τ e' soluzione dell'equazione ricorsiva. La dimostrazione si svolge in due parti:

- **Se $\tau_h(\text{fix}_\tau)(x) = y$:** Per il lemma 1, esiste una certa restrizione finita ϑ di fix_τ tale che $\tau_h(\vartheta)(x) = y$. Ma allora, ricordando come sono fatte le funzione f_n che compongono fix_τ , esistera' un certo n tale che $\vartheta \leq f_n$. Quindi, dal momento che ϑ e' restrizione finita di f_n e $\tau_h(\vartheta)(x)=y$, allora per definizione di restrizione finita avremo anche $\tau_h(f_n)(x) = (f_{n+1})(x) = y$. Ma sappiamo che fix_τ estende qualsiasi f_n , quindi anche f_{n+1} , per cui $(\text{fix}_\tau)(x) = y$.
  Ora, dal momento che la premessa da cui eravamo partiti era che $\tau_h(\text{fix}_\tau)(x) = y$, concludiamo che $\tau_h(\text{fix}_\tau) \leq \text{fix}_\tau$.

- **Se $\text{fix}_\tau(x) = y$:** Per definizione di fix_τ , esiste un n tale che $f_n(x)=y$. Ma allora abbiamo che $\tau_h(f_{n-1})(x) = (f_n)(x) = y$, da cui, applicando il lemma 1, deduciamo che esiste una restrizione finita ϑ di f_{n-1} tale che $\tau_h(\vartheta)(x)=y$. Quindi, dal momento che $f_{n-1} \leq \text{fix}_\tau$, allora anche $\vartheta \leq \text{fix}_\tau$ da cui, per definizione di restrizione finita, $\tau_h(\text{fix}_\tau)(x) = y$.
  Ora, dal momento che la premessa da cui eravamo partiti era che $\text{fix}_\tau(x) = y$, concludiamo che $\text{fix}_\tau \leq \tau_h(\text{fix}_\tau)$.

Unendo I due risultati trovati concludiamo che $\tau_h(\text{fix}_\tau) = \text{fix}_\tau$.

Dim (2):

Dobbiamo dimostrare che, se g e' un'altra soluzione dell'equazione ricorsiva, cioe' $\tau_h(g) = g$, allora $\text{fix}_\tau \leq g$. Per far questo, procediamo per induzione:

- **Caso base:** Essendo f_0 un programma per la funzione vuota, avremo sempre che $f_0 \leq g$.

- **Induzione:** Supponiamo per ipotesi induttiva che $f_{n-1} \leq g$.
  Ora, se per un certo x si ha che $f_n(x)=y$, allora, per definizione delle f_n , si ha $\tau_h(f_{n-1})(x) = f_n(x) = y$. Quindi, applicando a questa espressione il lemma 1, otteniamo che esiste una certa restrizione finita ϑ di f_{n-1} tale che $\tau_h(\vartheta)(x)=y$. Ma, per ipotesi induttiva, abbiamo $f_{n-1} \leq g$, per cui concludiamo che $\vartheta \leq g$. Quindi, dal momento che $\tau_h(\vartheta)(x)=y$, avremo anche che $\tau_h(g)(x)=y$ e, dal momento che per ipotesi del teorema sappiamo che g e' un punto fisso del funzionale, cioe' $\tau_h(g)=g$, allora concludiamo che $g(x)=y$. Visto che la premessa del discorso era che $f_n(x)=y$, allora $f_n \leq g$, quindi, avendo dimostrato questa proprieta' per ogni n, concludiamo che $\text{fix}_\tau \leq g$.

21.2 Qualche esempio di applicazione del 1° teorema di ricorsione

Vediamo adesso qualche esempio su come puo' essere utilizzato il primo teorema di ricorsione per risolvere alcune equazioni ricorsive. Il concetto fondamentale che sta sotto questo procedimento e' quello di ricavare, data un'equazione ricorsiva, il funzionale ad essa associata. A questo punto, la soluzione dell'equazione e' la funzione fix_τ, vista nel primo teorema di ricorsione, associata a tale funzionale, la quale viene ricavata semplicemente calcolando la successione di funzioni $f_0 \leq f_1 \leq \ldots \leq f_n$. Essendo fix_τ l'unione di queste funzioni concentriche, **quando troviamo una f_n della successione che non riusciamo ad espandere, essa e' fix_τ.**

Esempio 1:

Vogliamo trovare la soluzione della seguente equazione ricorsiva:

$$f(x) = \begin{cases} 3 & \text{se } x = 0 \\ f(f(x+1)) & \text{se } x > 0 \end{cases}$$

Sostituiamo ora la funzione f(x) con un programma y che la calcola:

$$\phi_y(x) = \begin{cases} 3 & \text{se } x = 0 \\ \phi_y(\phi_y(x+1)) & \text{se } x > 0 \end{cases}$$

Chiamiamo quest'ultima funzione $\tau(y,x)$. A quest'ultima, essendo calcolabile, possiamo applicare il teorema del parametro. Esiste quindi una funzione calcolabile e totale h(x) tale che:

$$\phi_{h(y)}(x) = \tau(y,x) = \begin{cases} 3 & \text{se } x = 0 \\ \phi_y(\phi_y(x+1)) & \text{se } x > 0 \end{cases}$$

Ora, dal momento che la codifica di f(x), cioe' y, appare solo come indice in questa definizione, allora h e' una funzione estensionale, per cui τ_h e' un funzionale ricorsivo. Ma dalla definizione di funzionale ricorsivo sappiamo che $\tau_h(f) = \Phi_{h(y)}$, per cui possiamo scrivere la seguente definizione del funzionale τ_h :

$$\tau_h(f)(x) = \begin{cases} 3 & \text{se } x = 0 \\ f(f(x+1)) & \text{se } x > 0 \end{cases}$$

A questo punto, come detto all'inizio del paragrafo, dobbiamo ricavare la successione di funzioni $f_0 \leq f_1 \leq \ldots \leq f_n$ vista nel primo teorema di ricorsione. Avremo:

$f_0 = f_\phi$

$$f_1 = \tau_h(f_0) = \begin{cases} 3 & \text{se } x = 0 \\ \uparrow & \text{se } x > 0 \end{cases}$$

Vediamo subito che $\tau_h(f_0) \neq f_0$, quindi, non abbiamo ancora trovato fix_τ. Dobbiamo quindi proseguire:

$$f_2 = \tau_h(f_1) = \begin{cases} 3 & \text{se } x = 0 \\ \uparrow & \text{se } x > 0 \end{cases}$$

Notiamo subito, invece, che $\tau_h(f_1) = f_1$, e questo indica che, non potendo espandere ulteriormente f_1, allora $f_1 = fix_\tau$, quindi f_1 e' soluzione dell'equazione ricorsiva. Per verificarlo basta sostituirla ad f(x) nella definizione iniziale.

Esempio 2:
Vogliamo trovare, analogamente a prima, la soluzione della seguente equazione ricorsiva:

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

Procedendo come nell'esempio 1, dopo aver sostituito ad f(x) un programma che la calcola, aver applicato alla funzione risultante il teorema del parametro e avere visto che la funzione h(y) che ci interessa e' estensionale, otteniamo la seguente definizione del funzionale associato all'equazione ricorsiva:

$$\tau_h(f)(x) = \begin{cases} 2 & \text{se } x = 2 \\ f(x+1) & \text{altrimenti} \end{cases}$$

Come fatto nell'esempio precedente, cerchiamo ora di calcolare la successione funzioni $f_0 \leq f_1 \leq \ldots \leq f_n$ fino a quando arriviamo ad una certa f_n non espandibile:

$f_0 = f_\phi$

$$f_1 = \tau_h(f_0) = \begin{cases} 2 & \text{se } x = 2 \\ \uparrow & \text{altrimenti} \end{cases}$$

Vediamo subito che $\tau_h(f_0) \neq f_0$, quindi, non abbiamo ancora trovato fix_τ. Dobbiamo quindi proseguire:

$$f_2 = \tau_h(f_1) = \begin{cases} 2 & \text{se } x = 2 \\ 2 & \text{se } x = 1 \\ \uparrow & \text{altrimenti} \end{cases}$$

Anche in questo caso vediamo che $\tau_h(f_1) \neq f_1$, quindi, non abbiamo ancora trovato fix_τ. Dobbiamo proseguire ulteriormente:

$$f_3 = \tau_h(f_2) = \begin{cases} 2 & \text{se } x = 2 \\ 2 & \text{se } x = 1 \\ 2 & \text{se } x = 0 \\ \uparrow & \text{altrimenti} \end{cases}$$

$$f_4 = \tau_h(f_3) = \begin{cases} 2 & \text{se } x = 2 \\ 2 & \text{se } x = 1 \\ 2 & \text{se } x = 0 \\ \uparrow & \text{altrimenti} \end{cases}$$

Finalmente vediamo che $\tau_h(f_3) = f_3$, quindi la successione non e' piu' espandibile, allora $f_3 = \mathbf{fix}_\tau$, quindi f_3 e' soluzione dell'equazione ricorsiva. Per rendersene conto basta sostituirla ad f(x) nella definizione iniziale.