Come risolvere un problema ricorsivo

Giacomo Bergami

3 marzo 2018

1

Una buona definizione ricorsiva di un problema $\mathcal{P}_{f,g}$ si poggia su due definizioni chiave, ovvero quella di "caso base" e "caso induttivo". Questo approccio viene utilizzato anche il logica ed in matematica discreta per fornire dimostrazioni[1] "strutturali" (se ben applicato, consente di ottenere dimostrazioni corrette, e quindi anche programmi corretti[2]).

- Si definisce **caso base** un dato d per il quale il programma riesce a fornire immediatamente una risposta, senza ulteriori chiamate ricorsive. Matematicamente:

$$\mathcal{P}_{f,g}(d) = f(d)$$

- Si definisce **caso induttivo** un dato d che può essere scomposto in due componenti, $\langle d_1, d_2 \rangle$, dove d_1 può essere valuato immediatamente mentre d_2 richiede una valutazione di $\mathcal{P}_{f,g}(d_2)$. La valutazione di $\mathcal{P}_{f,g}(d) \equiv \mathcal{P}_{f,g}(\langle d_1, d_2 \rangle)$ è data da una valutazione parziale di d_1, che deve essere combinata con quella restituita da $\mathcal{P}_{f,g}(d_2)$, sul quale si applica la ricorsione. Più matematicamente:

$$\mathcal{P}_{f,g}(\langle d_1, d_2 \rangle) = g(f(d_1), \mathcal{P}_{f,g}(d_2))$$

Come si può osservare, l'obiettivo dell'applicazione dell'induzione è quello di applicare la definizione ricorsiva di $\mathcal{P}_{f,g}$ solamente su una sotto-componente di d, ovvero d_2, in modo che si raggiunga sempre il caso base. In questo modo la "correttezza" garantisce anche che non si verifichino mai Stack-Overflow.

Esempio. *Ogni numero naturale* $\mathbb{N}$ *può essere descritto induttivamente tramite un caso base, zero* 0*, ed un qualsiasi numero successivo allo zero* $(n)+\mathbf{1}$ *Data questa definizione triviale di numero intero, potremmo pensare alla definizione del seguente algoritmo (inefficiente ma logicamente corretto): per la somma di due numeri* n *ed* m

- Se $n = 0$, allora $n + m = m$

[1] `https://it.wikipedia.org/wiki/Principio_d%27induzione`
[2] `http://www-sop.inria.fr/marelle/Laurent.Thery/formal/curry.html`

- Se $n = \tilde{d_2} + d_1$ con $d_1 = 1$, allora valuta $\tilde{n} + m$ e poi incrementa questo risultato di uno.

In questo caso avremo che il programma somma $\mathcal{P}_{f,g}(n, m)$ avrà come funzione f la costante 1 ($f(x) = 1$) e g la funzione somma così definita $g(f(1), \mathcal{P}_{f,g}(d_2, m)) = f(1) + \mathcal{P}_{f,g}(d_2, m) = \mathcal{P}_{f,g}(d_2, m) + \mathbf{1}$. Quindi avremo:

$$\mathcal{P}_{f,g}(n, m) = \begin{cases} m & n = 0 \\ \mathcal{P}_{f,g}(d_2, m) + \mathbf{1} & n = d_2 + \mathbf{1} \end{cases}$$

Osservate che la funzione restituisce un numero somma utilizzando la definizione induttiva di numero naturale.

1.1 Esercizio

A questo punto, proviamo a risolvere il problema **scrivere un programma con metodo ricorsivo che conti il numero di cifre dispari in un numero**. Proviamo quindi a decomporre questo problema in un problema ricorsivo che utilizza il principio d'induzione:

- Se il numero è negativo, lo converto in un numero positivo, così da non ripetere la procedura induttiva di analisi delle cifre sia per il caso negativo, sia per il caso positivo.
- Se il numero è positivo, procedo come utilizzando l'induzione:
  1. Se il numero è compresto tra 0 e 9 compresi, allora il problema si riduce a valutare se il numero è pari o dispari.

     $$\mathcal{P}_{f,g}(d) = f(d) \qquad f(d) = \texttt{(}d\,\texttt{\% 2 == 0 ? 0 : 1)}$$

  2. Altrimenti, mi posso chiedere se la cifra meno significativa sia dispari, quindi $d_1 = d\,\texttt{\%10}$. Se quest'ultima cifra è dispari, allora restituisco uno e zero altrimenti. Poiché $(d\,\texttt{\%10})\,\texttt{\%2} = d\,\texttt{\%2}$, allora abbiamo che $f(d_1) = \mathcal{P}_{f,g}(d_1)$.

     Da definizione precedente, abbiamo che $d_2 = d\,\texttt{/10}$, in quanto dobbiamo analizzare la parte rimanente del numero, escludendo la cifra meno significativa.

     Alla fine, sommiamo il numero di cifre dispari di d_2 con il valore di $f(d_1)$, e quindi g è la funzione somma: $g(x, y) = x + y$.

Otterremo quindi la seguente definizione matematica:

$$\mathcal{P}_{f,g}(d) = \begin{cases} \mathcal{P}_{f,g}(-d) & d < 0 \\ \texttt{(}d\,\texttt{\% 2 == 0 ? 0 : 1)} & 0 \leq d \leq 9 \\ f(d\,\texttt{\%10}) + \mathcal{P}_{f,g}(d\,\texttt{/10}) & d \geq 10 \end{cases}$$

o equivalentemente:

$$\mathcal{P}_{f,g}(d) = \begin{cases} \mathcal{P}_{f,g}(-d) & d < 0 \\ (d \,\texttt{\% 2 == 0 ? 0 : 1)} & 0 \leq d \leq 9 \\ \mathcal{P}_{f,g}(d \,\texttt{\%10}) + \mathcal{P}_{f,g}(d \,\texttt{/10}) & d \geq 10 \end{cases}$$

Si veda il sorgente di `CheckIfOdd.java` per una sua implementazione.

2 Tail Recursion

Una definizione ricorsiva come quella precedente, benché sia di immediata definizione, non è efficiente. Nella maggior parte dei casi, la soluzione ottimale è quella di non calcolare il risultato assieme alla chiamata ricorsiva, ma di passare a tale chiamata ricorsiva tale soluzione parziale r. Otteniamo quindi la seguente definizione:

$$\mathcal{P}'_{f,g}(d, r) = g(f(d), r)$$
$$\mathcal{P}'_{f,g}(\langle d_1, d_2 \rangle, r) = \mathcal{P}'_{f,g}(d_2, g(f(d_1), r))$$

Tuttavia, questo richiede che si conosca quale sia il valore $\overline{r}$ di r da fornire alla prima applicazione della funzione $\mathcal{P}'_{f,g}$. Quindi, otterremo un algoritmo ricorsivo che utilizza il principio di induzione nel seguente modo:

$$\mathcal{P}_{f,g}(d) = \mathcal{P}'_{f,g}(d, \overline{r})$$

2.1 Esercizio

Proviamo ora a riscrivere l'esercizio precedente utilizzando un approccio tail recursive. Nel nostro caso avremo che $\overline{r} = 0$, perché alla prima esecuzione non avremo la possibilità di analizzare nessuna cifra.

Conseguentemente, avremo:

$$\mathcal{P}_{f,g}(d) = \mathcal{P}'_{f,g}(d, 0)$$

$$\mathcal{P}'_{f,g}(d, r) = \begin{cases} \mathcal{P}'_{f,g}(-d, r) & d < 0 \\ r + (d \,\texttt{\% 2 == 0 ? 0 : 1)} & 0 \leq d \leq 9 \\ \mathcal{P}'_{f,g}(d \,\texttt{/10}, r + (d \,\texttt{\% 2 == 0 ? 0 : 1)}) & d \geq 10 \end{cases}$$

Si veda il sorgente di `CheckIfOddTailRecursive.java` per una sua implementazione.

Osserva. *È possibile applicare l'induzione anche in senso inverso, ovvero analizzando la prima cifra del numero intero invece dell'ultimo. Quest'ultimo richiede l'utilizzo della funzione logaritmo della classe* `Math`*. Viene lasciata per esercizio.*