Logica per l'informatica
Introduzione alla parte informatica

Andrea Malvezzi

28 Novembre, 2024

Contents

1 Introduzione

Gli informatici vogliono lavorare con tipi di dato di dimensioni finite ma unbounded, usando strutture ricorsive.
Considerando S come il **successore** di un numero naturale e O come lo zero, il numero naturale corrisponde a SO (successivo di zero), mentre il 3 si indica con la seguente sequenza ricorsiva: S(S(S(O))).
Un programma è una descrizione in un certo linguaggio di una sequenze logica capace di produrre un output a partire da un input.
Dire, nel codice, che:
0 + n = n corrispondea dire 0 + S(O) = S(O).
Oppure, di nuovo:
(S n) + m = S (n+m) corrisponde a dire S(S(O + (SO))) = S(S(SO)).
Il tutto supponendo che n sia pari ad 1 (S(O) equivale al successivo di zero, quindi uno).

2 Linguaggi funzionali

Nei linguaggi funzionali non si possono riassegnare variabili, quindi i loop non hanno senso, in quanto non terminerebbero mai. Tuttavia, trattando dati ricorsivi, si possono definire funzioni in grado di richiamare se stesse. Questo potrebbe fare divergere il programma (portarlo alla non-terminazione). Ad esempio:
con f(x) = f(S x) si avrebbe f(0) = f(S 0)=f(S(S 0))=f(S(S(S 0)))=...
Qui dire 0 + 100 impiegherebbe una sola operazione in quanto il primo numero equivale a zero, mentre per fare 100 + 0 occorrerrebbero 100 operazioni, in quanto aggiungerebbe un dito alla volta per contare ... Questo complica molto la dimostrazione delle proprietà delle operazioni.

3 I tipi di dati

Si usano Tipi di Dati Algebrici o **ADT**: questi possiedono un nome (caratteri, booleani, array etc...), eventuali parametri, una lista di possibili forme e costruttori (o un naturale è zero o è n volte il successore di (S O)). Ogni dato è quindi un albero dove ogni nodo corrisponde ad una possibile forma e tutti i suoi sotto-alberi sono altri possibili valori.
Ad esempio, i Booleani: $Bool = tt|ff$, con tt che rappresenta true e ff che

equivale a false;
Oppure le liste: $List = []|N : L$
Ad esempio: (S O) :: (O :: ((S(S O))::[])) è la lista che contiene in sequenza 1, 0 e infine 2. Come mai alla fine si mette ::[]? Perché per come abbiamo definiro l'ADT "List", si hanno due forme: o [], oppure N :: L (ovvero un numero seguito da una lista). Ciò significa che ogni numero deve essere seguito da una lista per corrispondere al tipo da noi definito.
Anche dire O:SO:[] sarebbe un errore, la forma corretta avrebbe le parentesi attorno a SO per indicare il valore di quella evalutazione, non l'evalutazione in sé. Quindi una cosa del tipo: O::(S O)::[].

4 Esempio di codice per definire un bool

inductive booleano : Type :=
    tt : booleano
    ff : booleano
Open booleano (è come se stessi importando il tipo booleano)

5 Esempio di codice per definire un numerico

inductive naturale : Type where
    O : naturale
    S : naturale -¿ naturale
Dove -¿ significa che deve prendere un naturale e deve ridarmi un naturale

6 Esempio di codice per definire una lista

inductive Lista : Type where
    empty : lista
    cons: naturale -¿ Lista -¿ Lista