Considerazioni su Tipi di Dati

Luca Abeni

April 19, 2017

Definire Tipi di Dato

- Molti linguaggi permettono di definire sinonimi per tipi di dato esistenti...
  - Non introducono nuovi valori
- La definizione di un nuovo tipo di dato richiede di definire nuovi valori
  - Tipo di dato: insieme di valori + operazioni su tali valori
- Caso più semplice: tipi enumerativi
  - Esempio: tipo `colore`, con valori `rosso`, `verde` e `blu`
- Nuovi valori o sinimimi di interi (a-la "`enum`" del C)?

Tipi Enumerativi

- Si *enumerano* esplicitamente tutti i possibili valori che una variabile del tipo può assumere (linguaggi imperativi) o che possono essere associati ad un simbolo (linguaggi funzionali)
- `enum colore = {rosso, verde, blu}`, `datatype colore = rosso | verde | blu`, ...
  - Prima della definizione, l'identificatore `rosso` non è riconosciuto dal linguaggio
  - Dopo la definizione, `rosso` identifica un valore di tipo `colore` (o è associato ad un intero???)
- Poi vanno definite le operazioni...
- Problema: numero di valori limitato

Generalizzando...

- Invece di enumerare valori costanti, costruttori
  - Costruttore di dato: mappa 0, 1 o più argomenti in un valore del tipo
  - "Scelta" fra più alternative (uno o più costruttori)
- Variante: insieme dei valori generati da un costruttore
  - Insieme di definizione: unione (disgiunta) delle varianti
- Sintassi ML:

```
datatype <name> = <cons1> [arg1]
                | <cons2> [arg2]
                | ...
                | <consn> [argn];
```

Esempio

```
datatype numero = intero of int | reale of real;

val sommanumeri = fn
        (intero a, intero b) => intero (a + b)
      | (intero a, reale  b) => reale  ((real a) + b)
      | (reale  a, intero b) => reale  (a + (real b))
      | (reale  a, reale  b) => reale  (a + b);

val sottrainumeri = fn
        (intero a, intero b) => intero (a - b)
      | (intero a, reale  b) => reale  ((real a) - b)
      | (reale  a, intero b) => reale  (a - (real b))
      | (reale  a, reale  b) => reale  (a - b);

     ...
```

Costruttori di Dato

- Costruttore senza argomenti (0 argomenti): valore costante
  - Genera un unico valore (identificato dal nome del costruttore)
    - Variante ad un unico valore
  - Tipo enumerativo: caso speciale, in cui tutti i costruttori non hanno argomenti
- Costruttore ad 1 o più argomenti: genera una variante del tipo
  - E' una funzione
  - Standard ML accetta un solo argomento $\rightarrow$ se servono più argomenti, tupla!

Ricorsione - 1

- Tecnica usata per definire un qualche tipo di “entità” basando la definizione sull’entità stessa che si sta definendo
  - “Entità”: funzione, insieme, tipo di dato...
- Basata sul concetto matematico di induzione
  - Definizione per casi (clausole)
  - Base induttiva: non fa riferimento all’entità che si sta definendo
  - Passo induttivo: permette di generare / calcolare nuovi valori a partire da valori precedentemente calcolati / generati / definiti

Ricorsione - 2

- Abbiamo già visto funzioni ricorsive...
  - Funzioni definite su numeri naturali
  - Base induttiva: valore della funzione per argomento più piccolo
  - Passo induttivo: definisce $f(n+1)$ basandosi su $f(n)$
- Ma la ricorsione si applica solo alla definizione funzioni???
  - La domanda fa presupporre che la risposta sia “no”...

Funzioni Ricorsive

- "Equivalente funzionale" dell'iterazione
- Principale utilizzo della ricorsione visto fino ad ora
- Funzione $f : \mathcal{N} \to \mathcal{X}$
  - Dominio: numeri naturali; Codominio: generico insieme $\mathcal{X}$
- Si definisce $f()$ tramite una funzione $g : \mathcal{X} \times \mathcal{N} \to \mathcal{X}$
  - Dominio: coppie composte da elementi di $\mathcal{X}$ e numeri naturali; Codominio: stesso codominio di $f()$
- Base induttiva: $f(0) = a$ (con $a \in \mathcal{X}$)
- Poi, $f(n+1) = g(n, f(n))$

Altri Esempi di Ricorsione / Induzione

- Induzione non solo per definire funzioni...
- Per esempio, definizione induttiva di insiemi
  - Base induttiva: si definisce un elemento appartenente all'insieme
  - Poi, si indica come generare elementi appartenenti all'insieme a partire da elementi dell'insieme stesso
- Esempio: numeri naturali $\leftarrow$ Peano
  - Base induttiva: 0 è un naturale ($0 \in \mathcal{N}$)
  - $n \in \mathcal{N} \Rightarrow n + 1 \in \mathcal{N}$
    - In altre parole, esiste una funzione $s : \mathcal{N} \rightarrow \mathcal{N} - \{0\}$

Ricorsione sui Dati

- Per definire un tipo di dato, ne va definito l'insieme dei valori (insieme di definizione)
- Come visto, un insieme può essere definito tramite induzione...
- Si può allora usare il meccanismo dell'induzione per definire un tipo di dato???
  - Induzione... Come?
  - Un costruttore può avere come parametro un valore del tipo di dato che si sta definendo!!! (passo induttivo)
  - Base induttiva: uno dei costruttori deve avere 0 parametri, o non avere parametri del tipo di dato che si sta definendo
- Chiaro, no???

Tipi di Dati Ricorsivi: Esempio

- Standard ML, definiamo i numeri naturali
- $0 \in \mathcal{N}$: costruttore `zero` (no argomenti)
- $n \in \mathcal{N} \Rightarrow n+1 \in \mathcal{N}$: costruttore `successivo`, ha come argomento un numero naturale

```
datatype naturale = zero | successivo of naturale;

val rec i2n = fn 0 => zero
              | x => successivo (i2n (x - 1));

val rec n2i = fn zero          => 0
              | successivo n => 1 + n2i n;

val rec nsum =
     fn zero          => (fn n => n)
     |  successivo a => (fn n => successivo (nsum a n));
```

Altro Esempio: Liste di Interi

- Una lista (per esempio, di interi) può essere definita ricorsivamente
  - La lista vuota è una lista (base induttiva)
  - Una lista non vuota è composta da un intero seguito da una lista (passo induttivo)
- Tipo ricorsivo: lista non vuota definita basandosi sul concetto di lista
- Due costruttori
  - Lista vuota
  - Lista con intero in testa

Liste di Interi

- Due costruttori
  - Costruttore di lista vuota (no argomenti)
  - Costruttore di lista non vuota (argomenti: numero intero e lista che lo segue)
- Altre operazioni
  - `car`: data lista non vuota, ritorna primo elemento (testa)
  - `cdr`: data lista non vuota, ritorna il resto della lista

Liste di Interi - Implementazione

- Tipi di dato ricorsivi direttamente supportati in linguaggi funzionali
  - Standard ML: `datatype`, due costruttori
  - Uno dei due costruttori ha argomento di tipo `int * lista`
- Linguaggi imperativi: puntatori!
  - Struttura con due campi
  - Campo valore e campo puntatore al prossimo elemento
  - Non potendo definire un tipo basandosi sul tipo stesso, si usa un puntatore (al tipo)!
- Ah... Ecco a cosa servono i puntatori!!!
- Vedere codice