Esercizi di Programmazione Curry

Marco Comini

17 aprile 2008

Gli esercizi seguenti servono per impratichirsi con le features tipiche/avanzate del linguaggio Curry. Per la parte "deterministica" di Curry valgono in buona sostanza esempi ed esercizi di Haskell, che vi invito a fare prima di prendere in considerazione quanto segue in quanto qui intendo focalizzarmi sulla parte che sfrutta la potenza espressiva delle variabili logiche trattando contemporaneamente programmi più "sostanziosi".

Si ricorda che in questo contesto un predicato è una funzione con risultato `Bool` e un constraint è una funzione con risultato `Success`.

Si scrivano i programmi con variabili anonime ove possibile.

1 Basics

Si cerchi di dare soluzioni che possano funzionare con queries con variabili logiche quando ha senso.

1. Scrivere un constraint che determina se una lista di elementi è palindroma (o le genera se chiamato con una variabile logica).

2. Si scriva un programma che data una lista ne restituisce una permutazione (tutte non-deterministicamente).

3. Si scriva un programma che dato un n restituisce una permutazione $[p_1, \ldots, p_n]$ della lista $[1, \ldots, n]$ in modo che, su una scacchiera di $n \times n$, n regine messe in posizione $(1, p_1), \ldots, (n, p_n)$ non si mangino fra loro.

4. Scrivere una funzione (non-deterministica) `treepath` che determina percorsi all'interno di un albero binario. Quindi, a titolo di esempio,

   ```
   treepath (Node 1 (Node 2 (Node 3 Void Void) Void) (Node 4 Void Void))
   ```

   restituisce 2 soluzioni: `[1,2,3]` e `[1,4]`. Non si escluda `treepath Void = []`.

   Si faccia attenzione a *non generare la stessa soluzione più volte.*

2 Soluzione di Problemi di Ricerca

Si ricordi che si dispone di varie funzioni per il non-determinismo nel Prelude, come

```
(?) :: a -> a -> a
```

per generare espressioni non-deterministiche.

1. Si scriva un programma per risolvere il gioco del Giro del Cavallo. Partendo da un angolo di una scacchiera $n \times n$ riuscire a muovere il cavallo in modo che passi su *ogni* casella *esattamente* una volta.

2. Abbiamo 2 recipienti di capacità B e S (interi), con $B > S$. I due recipienti possono essere riempiti da, o svuotati in, una riserva illimitata d'acqua. Un recipiente può essere versato nell'altro fino a che lui è vuoto o l'altro è pieno. Usando la seguente dichiarazione di tipo

   ```
   data Action = FillSmall | FillBig | PourSmall | PourBig
               | SmallInBig | BigInSmall
   ```

si scriva un programma per trovare una sequenza di azioni che lasciano una quantità C nel recipiente piccolo.

3. Si scriva una funzione Curry (non-deterministica) che fornisca una (ogni) rappresentazione di una (ogni) soluzione del gioco del Contadino con la Capra, i Cavoli e il Lupo. Il contadino deve riuscire a portare Capra, Cavoli e Lupo sulla riva opposta di un fiume con una barca che può contenere (oltre a lui) al massimo uno dei tre soggetti. Però se lascia soli Capra e Cavoli la Capra si mangerà i Cavoli e se lascia soli Lupo e Capra il Lupo si mangerà la Capra.

4. Preso un gruppo $r_1, \ldots, r_n$ di rettangoli si vuole determinare la *massima* sequenza di *miglior* incapsulamento, cioè la più lunga sequenza $r_{x_1}, \ldots, r_{x_k}$ per cui il rettangolo r_{x_i} è strettamente contenuto nel rettangolo $r_{x_{i+1}}$ (eventualmente ruotando i rettangoli di $\pi/2$) e *a parità di lunghezza* quella che minimizza la somma $\sum_i (h_{x_{i+1}} - h_{x_i})(l_{x_{i+1}} - l_{x_i})$, dove h e l sono le altezze e larghezze dei rettangoli (per come si ritrovano dopo averli eventualmente girati).

   Si scriva una funzione che data una lista di coppie che rappresenta i rettangoli determina la (una) massima sequenza di miglior incapsulamento.

5. Si vuole risolvere il seguente problema: dati certi numeri positivi distinti ed un valore positivo v determinare se esiste un'espressione aritmetica che valuti a v in cui ogni numero compaia *esattamente* una volta. Ad esempio una soluzione per $[2, 3, 6, 8]$ e 24 è $(2 + 8) * 3 - 6$.

   Codificando le espressioni aritmetiche mediante il tipo di dato

```
data Exp = N Int | O Op Exp Exp
data Op = Add | Mul | Sub | Div
```

   si scriva una funzione (non-deterministica) *solve* che data una lista di numeri e un valore restituisce una soluzione (se esiste).

6. Ci sono quattro numeri (1, 3, 7, 11) che devono attraversare un ponte stretto di notte. Sul ponte non riescono a passare più di 2 numeri alla volta. I numeri hanno a disposizione una sola torcia per vedere al buio, quindi i numeri all'andata passano in coppia e poi uno di quelli che si trovano sull'altra sponda torna indietro con la torcia per permettere di passare anche agli altri con la luce.

   Il costo del viaggio di andata è la somma dei due numeri. Il costo del viaggio di ritorno è il valore del numero che torna indietro.

   Il costo totale di un attraversamento di tutti i numeri all'altro lato del ponte è semplicemente dato dalla somma dei costi dei singoli viaggi.

   Si scriva una funzione Curry (non-deterministica) che dato un costo c restituisca una soluzione (tutte) a costo totale minore di c. Con soluzione si intende una lista che codifichi (in qualche modo) le configurazioni dei numeri rispetto al ponte.

7. Si ha un grafo con archi orientati etichettati da un colore e i cui nodi sono identificati da un numero univoco e sono, a loro volta, etichettati con un colore (non necessariamente distinto). Si parte con 2 nodi "marcati" ed un nodo "goal". La marcatura può essere spostata lungo un arco se il nodo destinazione non è già marcato e se il colore dell'arco è lo stesso del nodo attualmente marcato. Bisogna trovare una sequenza di azioni che fanno arrivare una marcatura nel nodo goal.

   Il grafo viene rappresentato mediante una lista di archi e una funzione che ad ogni identificativo di nodo ne associa il colore secondo le definizioni

```
data Graph = Graph [Arrow] Coloring
data Arrow = Arrow Node Node Color
type Node = Int
type Coloring = Node->Color
data Color = Red | Blue | Green | Magenta | ...
```

   Si scriva una funzione Curry (non-deterministica) che preso un grafo e tre identificatori (distinti) di nodo, il goal e i due iniziali, fornisca una (ogni) rappresentazione di una (ogni) soluzione.