Programmazione Funzionale – Febbraio 2017

Nota: è indispensabile specificare il tipo e dare una descrizione dichiarativa di ogni funzione ausiliaria utilizzata (anche locale), altrimenti non verrà presa in considerazione (ad eccezione delle funzioni il cui tipo e specifica sono già dati nel testo).

Un elemento del tipo `'a option` si può considerare come un pattern per valori di tipo `'a`: `Some x` è conforme soltanto al valore `x`, mentre `None` è conforme a ogni valore di tipo `'a` (si comporta come la variabile muta nel pattern matching di OCaml).

Sia data la seguente dichiarazione di tipo per la rappresentazione di grafi mediante liste di archi:

```
type 'a graph = ('a * 'a) list
```

.

Siano `g` un `'a graph`, `pattern_list` una `'a option list` di lunghezza k e `path: 'a list` un cammino nel grafo `g`. Diciamo che `path` è *inizialmente conforme* a `pattern_list` se `path` ha almeno k elementi e ciascuno dei suoi primi k elementi è conforme all'elemento di `pattern_list` che si trova nella stessa posizione. Inoltre `path` è *parzialmente aciclico* rispetto a `pattern_list` se nessun nodo successivo al k-esimo occorre più di una volta nel cammino.

Ad esempio, il cammino `[1;3;4;3;4;2]` nel grafo rappresentato in basso è inizialmente conforme alla lista `[None;Some 3;None;Some 3;None]`, ed è anche parzialmente aciclico rispetto ad essa (la lista ha 5 elementi, e l'unico elemento del cammino successivo al quinto è 2, che occorre una sola volta nel cammino). Lo stesso cammino è anche inizialmente conforme alla lista `[None;Some 3;None;Some 3]`, ma non è parzialmente aciclico rispetto a questa lista, in quanto la lista ha 4 elementi, ed il quinto nodo del cammino (4) occorre 2 volte nel cammino stesso. Qualsiasi cammino nel grafo è inizialmente conforme alla lista vuota e, se non contiene cicli, è anche parzialmente aciclico rispetto ad essa.

Scrivere una funzione

whichpath: 'a graph → 'a option list → 'a → 'a → 'a list ,

tale che `whichpath g pattern_list start goal` riporti, se esiste, un cammino da `start` a `goal` in `g` che sia inizialmente conforme a `pattern_list` e parzialmente aciclico rispetto ad essa. Se un tale cammino non esiste, la funzione solleverà un'eccezione.

Si osservi che la ricerca del cammino dovrà seguire la lista di pattern fino a che non è vuota, ed il controllo di cicli dovrà avvenire soltanto quando tale lista è vuota (ma controllando comunque il nodo visitato rispetto a tutti i nodi del cammino). Nel caso in cui la lista di pattern sia già inizialmente vuota, il risultato deve essere semplicemente un cammino senza cicli da `start` a `goal`.

Ad esempio, se `g` è una rappresentazione del grafo della figura:

- `whichpath g [None;None;None;None;None] 1 5` può riportare il cammino [1;2;5;5;5] oppure [1;3;4;2;5].
- `whichpath g [Some 1;None;Some 4;Some 3;None] 1 5` riporterà il cammino [1;3;4;3;4;2;5].
- `whichpath g [None;None;Some 3] 1 4` fallisce, perché dal nodo 1 non si può raggiungere il nodo 3 con un cammino che abbia esattamente 3 nodi.
- `whichpath g [None;None;None;None] 1 4` fallisce, perché la parte iniziale di ogni cammino dal nodo 1 al nodo 4 è necessariamente [1;3;4], e se il cammino deve avere almeno 4 nodi (come richiesto dalla lista di pattern), il quarto nodo del cammino sarà necessariamente di nuovo 3, che già occorre nel cammino stesso.
- `whichpath g [Some 1;None;Some 4;Some 3] 1 5` fallisce, perché la parte iniziale di qualsiasi cammino inizialmente conforme alla lista deve necessariamente essere [1;3;4;3] e, per arrivare da 3 a 5 si deve necessariamente ripassare per il nodo 4, che già occorre precedentemente nel cammino.