Programmazione Funzionale – Settembre 2022

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).

Supponiamo di avere un insieme di elementi di dimensioni diverse e che ciò si rappresenti mediante una *mappa delle dimensioni*, cioè una lista di tipo

`('a list * 'b) list`,

dove ogni coppia `(lista,d)` della mappa rappresenta il fatto che gli elementi di `lista` hanno dimensione `d`. Ad esempio, la lista

`[([1;2;3;4],"piccolo");([5;6;7;8],"medio");([9;10;11;12],"grande")]`

rappresenta il fatto che 1,2,3,4 hanno dimensione "piccolo", 5,6,7,8 hanno dimensione "medio" e 9,10,11,12 hanno dimensione "grande".

1. Scrivere una funzione `dim: 'a → ('a list * 'b) list → 'b` tale che `dim x dim_map` riporti la dimensione di `x` secondo `dim_map`. La funzione solleverà un'eccezione se `dim_map` non associa a `x` alcuna dimensione.

   Ad esempio, se `dim_map=[([1;2;3;4],"piccolo");([5;6;7;8],"medio"); ([9;10;11;12],"grande")]`, `dim 6 dim_map` riporterà `"medio"`, mentre `dim 100 dim_map` solleverà un'eccezione.

2. Definire una funzione

   `good_dim: ('a list * 'b) list → 'a → 'b list → bool`,

   tale che `good_dim dim_map x dimlist = true` se la dimensione di `x` secondo `dim_map` è un elemento di `dimlist`. In tutti gli altri casi, cioè se la dimensione di `x` non appartiene a `dimlist` oppure se `dim_map` non associa a `x` alcuna dimensione, la funzione riporta `false`. La funzione non deve mai sollevare eccezioni.

3. Rappresentiamo i grafi orientati mediante liste di successori:

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

   Scrivere una funzione `path` di tipo

   `('a list * 'b) list → 'b list → 'a graph → 'a → 'a → 'a list`

   tale che `path dim_map dimlist g start goal` riporti, se esiste, un cammino nel grafo `g` dal nodo `start` al nodo `goal` che contenga soltanto nodi la cui dimensione, secondo la lista `dim_map`, è un elemento di `dimlist`. **Attenzione:** la funzione deve fallire solo se non esiste un cammino con questa caratteristica (e non se, durante la ricerca del cammino, si visita un nodo "senza dimensione").

   Se ad esempio `dim_map=[([1;2;3;4],"piccolo");([5;6;7;8],"medio"); ([9;10;11;12],"grande")]` e `dimlist=["grande"; "piccolo"]`, il cammino non potrà contenere nodi diversi da 1,2,3,4,9,10,11,12. Ad esempio, se `g = [(1,100);(1,5);(1,3);(3,1);(100,10);(3,10);(5,10)]`, `path dim_map ["grande";"piccolo"] g 1 10` riporterà il cammino `[1; 3; 10]`, mentre se `g = [(1,100);(1,5);(5,1);(100,10); (5,10)]`, `path dim_map ["grande"; "piccolo"] g 1 10` fallisce.

   Suggerimento: adattare l'algoritmo di ricerca di cammini in un grafo aggiungendo un caso di fallimento quando si visita un nodo la cui dimensione non è "buona" (vedi punto 2).