Programmazione Funzionale

Esercitazione 8 – Rappresentazione di Insiemi

Un insieme finito $A = \{a_1, \ldots, a_n\}$ puo essere rappresentato come:

- una lista `[a1;...;an]` eventualmente in un ordine diverso che non contiene ripetizioni. Ad esempio l'insieme $\{0, 2, 4\}$ puo essere rappresentato (senza ripetizioni) dalle liste `[0;2;4]` e `[2;0;4]`.
- una lista `[a1;...;an]` eventualmente in un ordine diverso che contiene ripetizioni. Ad esempio l'insieme $\{0, 2, 4\}$ puo essere rappresentato dalle liste `[0;2;4]` e `[2;2;0;4;0;0;0]`.

  N.B. Qualsiasi lista `lst` e la rappresentazione (con eventuale ripetizioni) di un unico insieme. Ma ad un insieme corrispondono tante rappresentazione come lista.
- Come una funzione `f:a' -> bool` che prende un elemento di tipo `a'` e ritorno `true` se e solo se l'elemento e contenuto nel insieme A.

Esercizio 1. Vogliamo implementare un modo di determinare se due liste sono uguale dopo riordinamento. Cioé se contengono i stessi elementi ma in un ordine diverso.

1. Definire una funzione `remove : a' -> a' list -> a' list` che prende un elemente `elem` di tipo `a'` e una lista `lst` di elementi di tipo `a'` e restituisce una lista `lstOut` tale che:
   - Se `elem` occore in `lst` allora `lstOut` corrisponde alla lista in entrata `lst` da cui e stato tolta la prima occorenza di `elem`.
   - Se `elem` non occore in `lst` allora `lstOut` corrisponde a `lst`.

   Ad esempio, `remove 2 [1;2;3;2]` restituisce `[1;3;2]` invece `remove 0 [1;2;3;2]` restituisce `[1;2;3;2]`.
2. Definire una funzione `removers : a' list -> (a' list -> a' list) list` che prende una lista `[a1;...;an]` e restituisce la lista di funzione `[remove a1 ; ... ; remove an]`.
3. Definire una funzione `apply : ((a-> b) list * a) -> b` che prende una lista di funzioni `[f1;...;fn]` e une elemento `elem` di tipo `a` e restituisce `fn(...(f1(elem))...)`.
4. Usando `removers` e `apply` definire una funzione `ordEqual: a' list * a' list -> bool` che prende due liste e determina se contengono i stessi elementi.
5. Cosa restituisce la funzione precedente `ordEqual` quando e applicata a due liste `la,lb` che contengono i stessi elementi ma con ripetizioni:
   - Nel caso in cui le ripetizioni sono le stesse cioe un elemento occore n volte in `la`, se e solo se occore n volte in `lb`. Per esempio `[2;1;2;3]` e `[3;1;2;2]`.
   - Senza nessuna condizione cioé `la` e `lb` contengono i stessi elementi ma con ripetizioni eventualmente diverse. Per esempio `[2;1;3;3]` e `[2;2;3;2;1]`.

Esercizio 2. Dato una lista `lst` e A l'insieme che rappresenta vogliamo costruire la funzione che rappresenta l'insieme A a partire di `lst`.

1. Qual l'insieme rappresentato dalle seguente liste; `[0;1;2;2;2]`, `[3;4;1]`, `[3;3;3;3;3]`, `[2;0;0;1]`.
2. Definire una funzione `charach : a' list -> (a' -> bool)` che prende una lista che rappresenta un insieme A e restituisce la funzione che rappresenta A, cioé la funzione ritorna `true` quando l'elemento appartiene `lst` e `false` altrimenti.

   Fate due versione di `charach`;
   - Una senza usare `List.mem`.
   - Una che usa `List.mem`.

Esercizio 3. La differenza simmetrica di due insiemi $X_1 \subseteq X$ e $X_2 \subseteq X$ corrisponde a l'insieme

$$X_1 \Delta X_2 \triangleq \{x \in X \mid (x \in X_1 \text{ and } x \notin X_2) \text{ or } (x \notin X_1 \text{ and } x \in X_2)\}$$

Dal punto di vista dei booleani la differenza simmetrica e collegata a l'operatore `XOR:bool*bool -> bool` che prende due booleani `b1` e `b2` e ritorna `true` se e solo se, uno dei booleani e falso mentre l'altro e vero.

Cerchiamo di implementare la differenza simmetrica sulle rappresentazione di un insieme.

1. Definire un funzione `diffSim: a' list * a' list -> a' list` che prende due liste `la` `lb` e restituisce la lista che corrisponde alla differenza simmetrica dei due insieme rappresentato da `la` e `lb`

2. Implementare la funzione `xor:bool*bool -> bool`.

3. Usando la funzione `xor` definire la funzione `diffSimf : (a'-> bool) * (a'-> bool) -> (a' -> bool)` che prende due funzione `fa: a' -> bool` e `fb: b' -> bool` e restituisce la funzione che corrisponde alla differenza simmetrica dei due insieme rappresentato da `fa` e `fb`.