Programmazione Funzionale

Esercitazione 12 – Backtracking sulle liste

Chiarifichiamo la terminologià;

- Una *sottolista* di una lista `lst = [a1;...;an]` e una lista della forma $[a_{u1};\ldots;a_{uk}]$ dove pe ogni $1 \leq i \leq k$ l'indice u_i e piu piccolo di u_{i+1} cioè $u_i < u_{i+1}$.

  Quindi e una sottolista di `lst` che rispetta il senso di lettura di `lst` e che non autorizza ripetizioni di elementi, ma eventualmente un elemento puo non occorere.

  Ad esempio `[2;3;5]` e una sottolista di `lst = [1;1;2;2;3;1;1;5]`, perche i elementi 2, 3 e 5 occorono in `lst` e in quel ordine di letture.

  Al contrario `[2;3;5]` non e una sottolista di `la = [5;2;4;4;3]` perche anche se 2, 3 e 5 occorono in `la` non occorono in quel ordine di lettura.

- Un *iterazione* di un elemento `a` e una lista che contiene solo l'elemento `a` cioè se non e vuota e della forma `[a ; ... ; a]`. Scriviamo a^n l'iterazione di `a` di lunghezza `a` (se $n = 0$ corrisponde alla lista vuota).
- Una *sottolista con ripetizioni* di un lista `[a1;...;an]` e una lista della forma $a1^{k1}@\ldots@an^{kn}$.

Esercizio 1. Cerchiamo in una lista una sottolista che verifica la proprietà P, tale P(`lst`) vale se e solo se il suo primo elemento e il l'ultimo elemento di `lst` e 1.

1. Definire una funzione `safetest : int -> 'a -> 'a list -> bool` che prende un intero `n` un elemento `a` e una lista `lst` e restituisce `true` se la lista contiene al `n`–esimo posto l'elemento `a` e false altrimenti, la funzione non solleva mai eccezione.
2. Definire la funzione `accept : 'a list -> bool` che restituisce true se e solo se la lista rispetta la proprietà.
3. Definire la funzione `reject : 'a list -> bool` che prende `lst` erestituisce true se e solo se ogni lista che contiene `lst` non verifica la propietà P.
4. Usando le funzioni `reject` e `accept` definire una funzione `backtrack : 'a list -> 'a list` che prende una lista e restituisce una sua sottolista che verifica la proprietà P. Si può usare una versione con un accumulatore. Se non e trovata nessuna sottolista la funzione sollevera un eccezione.
5. Usando `backtrack` e una dichiarazione `try ... with ...` Definire una funzione `resolvable :'a list -> bool` che prende una lista `lst` e restituisce `true` se e solo se `backtrack` ha trovato una sottolista con la proprietà P.
6. Definire `resolvable` senza usare `backtrack`, cioè quandè che una lista può contenire una sottolista che ha come primo e ultimo elemento 1?

Esercizio 2. Cerchiamo in una lista una sottolista *con ripetizioni* che rispetta la proprietà P tale che; P(`lst`) vale se e solo se la somma dei elementi di `lst` vale 5.

1. Definire la funzione `accept : 'a list -> bool` che restituisce true se e solo se la lista rispetta la proprietà.
2. Definire la funzione `reject : 'a list -> bool` che prende `lst` erestituisce true se e solo se ogni lista che contiene `lst` non verifica la propietà P.
3. Usando le funzioni `reject` e `accept` definire una funzione `backtrack : 'a list -> 'a list` che prende una lista e restituisce una sua sottolista con ripetizioni che verifica la proprietà P.

   Si può usare una versione con un accumulatore. Se non e trovata nessuna sottolista la funzione sollevera un eccezione.
4. Adattare il problema aggiungendo un parametro n cioè P(`lst`, n) vale se e solo se la somma dei elementi di `lst` vale n, definire `accept`, `reject` e `backtrack` con un parametro in piu.

Esercizio 3. Immaginiamo il gioco seguente con un giocatore; il giocatore lancia un dado un numero finito di volte e inserisce in una lista il risultato dei suoi lanci. Ad esempio la lista `[2;3;5]` significa che il giocatore ha fatto tre tiri, ha tirato il dado e ha fatto 2 poi ha fatto 3 e poi 5.

La condizione di vincita e la seguente; il giocatore deve aver fatto un 1 seguito da un 2 nei prossimi due tiri.

Ad esempio `[1;4;3;2;3]` non e vincente perche il valore 2 non occore nei due prossimi tiri, invece `[1;3;2;8]` e vincente perche l'evento 2 occore al secondo tiro dopo 1.

Possiamo tradure questa condizione per vincere come una condizione P sulle liste che vale se e solo se il primo elemento della lista e 1, l'ultimo elemento e 2 e la lunghezza della lista e inferiore o uguale a 3.

1. Definire la funzione `accept : 'a list -> bool` che restituisce true se e solo se la lista rispetta la proprietà.

2. Definire la funzione `reject : 'a list -> bool` che prende `lst` erestituisce true se e solo se ogni lista che contiene `lst` non verifica la propietà P.
3. Usando le funzioni `reject` e `accept` definire una funzione `backtrack : 'a list -> 'a list` che prende una lista e restituisce una sua sottolista (senza ripetizioni) che verifica la proprietà P.

   Si può usare una versione con un accumulatore. Se non e trovata nessuna sottolista la funzione sollevera un eccezione.