Università di Pisa
Corso di Programmazione II (273AA)

Progetto Ocaml - Relazione finale

Alessandro Antonelli
(matricola 507264, corso A)

Sessione estiva-autunnale A.A. 2018/2019
Appello settembre 2019

1 Come eseguire il codice e i test

Avviare un'istanza di un interprete interattivo Ocaml e copiare ed incollare al suo interno l'intero contenuto del file `EvalExtended.ml` e poi del file `test.ml`. Dovrebbe comparire la scritta "tutti i test superati".

Il programma è stato sviluppato e testato tramite la versione 4.07.1 dell'interprete disponibile nella shell Linux (comando `ocaml` del terminale), su Ubuntu 18.04 LTS 64 bit.

2 Descrizione

Rispetto all'interprete fornito come base il sistema dei tipi è stato modificato aggiungendo nel tipo `exp` la produzione per gli alberi (`ETree`), per ApplyOver e Select; tra i valori esprimibili sono stati aggiunti gli alberi (`EvTree`, che si differenziano dalle espressioni-albero per avere il contenuto dei nodi di tipo `evT` e non `exp`). Inoltre la funzione `typecheck` è stata estesa per riconoscere alberi e valori funzionali.

Nella **valutazione degli alberi**, all'interno della `eval`, viene innanzitutto controllato che non ci siano *tag* ripetuti (per mezzo della funzione `tagSonoUnivoci`): se ce ne sono viene segnalato un errore. Successivamente viene eseguito il parsing dell'espressione-albero di input, eseguendo chiamate ricorsive alla `eval` per valutare l'espressione contenuta in ciascun nodo e i sottoalberi sinistro e destro; infine l'albero viene restituito come valore esprimibile.

La **Select** è stata implementata come funzione ricorsiva che effettua una visita dell'albero: per ogni nodo, se il tag corrisponde restituisce il nodo, altrimenti prosegue la ricerca ricorsivamente sui sottoalberi.

La **ApplyOver** è stata implementata come funzione ricorsiva che prende l'albero e la lista, e restituisce un nuovo albero con contenuto dei nodi frutto dell'applicazione della funzione. La verifica dell'appartenenza del nodo a quelli da modificare viene effettuata con una funzione d'appoggio `contains`. In caso positivo, viene costruita l'invocazione della funzione con il contenuto del nodo come parametro, e viene eseguita invocando la `eval` (per tale ragione, le due funzioni sono dichiarate come mutuamente ricorsive). Calcolato il valore risultante, viene costruito il nuovo nodo con tag immutato, valore pari a quanto calcolato, e sottoalberi dati dalle chiamate ricorsive. L'implementazione fornita permette di usare la ApplyOver anche con funzioni ricorsive (l'espressione con la ApplyOver deve comparire come corpo della `Let_rec` della funzione ricorsiva desiderata).