Relazione Progetto 2017/18
Programmazione II

Setzu Simone [508366]

2018/02/05

2 OCaml

Nella realizzazione del progetto si è scelto di estendere la sintassi astratta dell'interpetre con i comandi, il tipo commands, che non è altro che una lista, rappresenta i comandi ed è così strutturato:

- Code<commands>, rappresenta il codice;
- Command<exp,Command>, contiene un espressione e il prossimo comando da eseguire;
- None, comando vuoto;

Per poter scorrere la lista di comandi è stato necessario modificare la funzione di valutazione della semantica sem, facendo in modo di restituire il nuovo ambiente da passare al comando successivo.
Con l'introduzione dei comandi si è scelto di inserire due modalità di dichiarazione:

- Let<ide,exp>, dichiarazione nell'ambiente globale dell'espressione exp legata all'identificatore ide;
- Let in<ide,exp1,exp2>, dichiarazione dell'identificatore ide legato all'espressione exp1 nel blocco dell'espressione exp2

Un ulteriore variazione è stata quella di valutare le funzioni all'atto della dichiarazione, risolvendo i riferimenti non locali ed effettuando le operazioni con i riferimenti risolti, compilando di fatto il corpo della funzione.
Per fare ciò è stato necessario introdurre una funzione che porta gli elementi valutati in elementi non valutati, in maniera tale da poterli valutare successivamente con il parametro della funzione.

2.1 Alberi

Si è esteso l'interpetre con il tipo ETree, rappresentato come un albero che ha elementi di tipo tree, strutturati come segue :

- Empty;
- Node<Chiave,Valore,<tree >,<tree >>.

Il tipo di dato exp è esteso con i costrutti:

- Tree, valutazione di un espressione Tree;
- ApplayOver of exp * exp, applicazione di una funzione a tutti i valori di un Tree;
- Update of (d_Ide list) * exp * exp, applicazione di una funzione a tutti i valori di un Tree appartenenti al path (d_Ide list);
- Select of (d_Ide list) * exp * exp, selezione condizionale di un nodo appartenente al path (d_Ide list).

La semantica corrispondente è:

- ETree, viene valutata la semantica di tutti i suoi elementi nell'ambiente corrente.
- Si valutano entrambi i parametri nell'ambiente corrente. Dopo la valutazione il primo deve essere di tipo EFun, il secondo di tipo ETree. Poi applico ricorsivamente la funzione valutata nell'albero valutato facendo il binding tra l'ambiente presente al momento della dichiarazione della funzione e la coppia <parametro formale, parametro attuale>.
- Si valutano entrambi i parametri nell'ambiente corrente. Dopo la valutazione il primo deve essere di tipo EFun, il secondo di tipo ETree. Poi applico ricorsivamente la funzione applytree, visitando l'albero e controllando se il nodo valutato appartiene al primo elemento della lista passata come parametro, se questo è verificato chiamo ricorsivamente la funzione sui nodi figli passandogli il resto della lista.
- Si valutano entrambi i parametri nell'ambiente corrente. Dopo la valutazione il primo deve essere di tipo EFun, il secondo di tipo ETree. Poi applico ricorsivamente la funzione applytree che prende in ingresso un albero valutato, una funzione valutata e una lista. Scorro albero e lista e nel caso in cui trovo che il nodo valutato appartiene al primo elemento della lista passata come parametro applico la funzione valutata, controllando che il risultato sia un booleano, se il risultato è true restituisco il nodo altrimenti applico applico applytree prima al figlio sinistro e poi al figlio destro.