Progetto Programmazione 2

Parte 2: Ocaml

Autore: Francesco Buti

Data consegna: 7/01/2019

Richieste soddisfatte:

1. Definire la sintassi concreta del linguaggio e la sintassi astratta tramite una opportuna definizione di tipo in OCaML.
2. Definire l'interprete OCaMl del linguaggio funzionale assumendo la regola di scoping statico.
3. Verificare la correttezza dell'interprete progettando ed eseguendo una quantità di casi di test sufficiente a testare tutti gli operatori aggiuntivi.
4. Estendere il linguaggio con un nuovo operatore rt_eval(exp) che prende in ingresso una espressione e restituisce il valore ottenuto eseguendo exp con la regola di scoping dinamico.
5. Creare una batteria di test in grado di verificare il corretto funzionamento del programma

Funzionalità previste:

Il progetto prevede l'estensione del linguaggio didattico volto a includere la possibilità di creare elementi di tipo Dictionary, realizzata mediante una lista di identificatore*valore, e le relative funzioni DictGet, DictRm, DictClear, DictAdd, ApplyOver le cui funzionalità sono descritte nel testo del progetto allegato.

Per far ciò si rende necessaria una differenziazione tra la prima e la seconda versione, rispettivamente volte a creare un interprete che operi con scoping statico e scoping dinamico.

Funzioni ausiliarie:

- La prima versione prevede l'esistenza di due funzioni, funeval e funcalleval, necessarie per gestire le applicazioni di funzione su evT (anziché exp) e per applicare una funzione agli elementi di un dizionario. Tali funzioni ausiliarie si rendono necessarie al momento della chiamata di ApplyOver.
- La seconda versione prevede l'utilizzo di una funzione apply, in quanto lo scoping dinamico si rende incompatibile con le funzioni precedentemente citate. Per far ciò, si fa uso della valutazione della funzione (passata sotto forma di exp) per applicarla al secondo elemento di ogni coppia che compone la lista (ovvero il dizionario), e lasciare pressoché invariata la ApplyOver.

Note particolari:

Nell'implementazione adottata, i parametri da passare alle funzioni prevendono anche l'utilizzo degli identificatori, per rendere l'esperienza più semplice all'utente finale. Questo comporta chiaramente delle scelte implementative che possono risultare scomode (vedere il pattern matching della ApplyOver), ma si ritengono necessarie per creare una batteria di test notevolmente semplificata.

I test sono opportunamente progettati per il buon fine, per provarne il corretto funzionamento si incoraggia il lettore a cambiare i parametri e verificare le eccezioni lanciate in caso di errore