SECONDO PROGETTO PROGRAMMAZIONE 2

Università di Pisa, A.A. 2019-2020

Maria Vitali, 548154

Overview

Il progetto prevede la progettazione e realizzazione di una semplice estensione del linguaggio didattico funzionale presentato a lezione che permetta di manipolare *dizionari*.
Un *dizionario* è una collezione di valori identificati univocamente da una chiave, quindi può essere visto come una collezione di coppie chiave-valore dove la chiave è unica.
Il linguaggio di supporto utilizzato è OCaml.
Le varie componenti del progetto, presenti nel file .zip allegato, sono le seguenti:

- Interprete.ml
- Test.ml
- TypeChecker.ml

L' ultimo file, di supporto alla progettazione, presente nella cartella compressa è:

- RegoleOperazionali.pdf

Implementazione

Al linguaggio funzionale didattico sono stati aggiunti:

- il tipo di dato dizionario `Dict of pairlist` (dove `pairlist` è anch'esso un nuovo tipo)
  e le operazioni primitive per manipolarlo `Insert`, `Delete`, `Has_Key`, `Iterate`, `Fold`, `Filter`.
  Per le *regole operazionali* di questi nuovi costrutti si rimanda al file RegoleOperazionali.pdf
- alcune funzioni di supporto per la realizzazione delle operazioni introdotte e per controllare eventuali proprietà dei dati: `listMember`, `dictMember`, `evalPairlist`, `deleteFromDict`, `filter`, `iter`, `accFun`.

Nota: per permettere l'implementazione dell'operazione primitiva Fold, si è ritenuto necessario aggiungere i costrutti primitivi per la dichiarazione e l' applicazione di funzioni che prendono 2 parametri, rispettivamente
`BinFun of ide * ide * exp` , `BinFunCall of exp * exp * exp`

TypeChecker

Nel file TypeChecker.ml è stato implementato un semplice TypeChecker per il linguaggio funzionale esteso con il tipo *dizionario* e le operazioni per manipolarlo.
Il tipo tval introduce i possibili tipi riconosciuti dal linguaggio.
La funzione `teval : exp -> tenv -> tval = <fun>`
restituisce il tipo tval di ogni costrutto primitivo e riconosce eventuali errori di tipo.

Test: HOWTO

Per testare l'interprete (o il TypeChecker):

1. aprire un qualsiasi terminale
2. spostarsi all'interno della directory dove sono stati estratti i file del progetto
3. avviare l' interactive shell di OCaml:

   ```
   ~$ ocaml
   ```

   Nota: se non si ha installato OCaml, i passi 1 e 3 possono essere eseguiti semplicemente collegandosi in rete al sito https://try.ocamlpro.com/.
   Seguire poi le informazioni del sito per il caricamento dei file.
4. compilare ed eseguire l'interprete (o il TypeChecker) in questo modo:

   ```
   # #use "Interprete.ml";;
   ```

5. eseguire i test:

   ```
   # #use "Test.ml";;
   ```