Relazione Secondo Progetto Pr2, Fulvio Denza 544006

Il progetto si basa sull'interprete esposto a lezione e, in addizione, è stato inserito il dizionario con le rispettive operazioni: aggiunta di elemento, eliminazione di un elemento, svuotamento di dizionario, creazione di un nuovo dizionario e applicazione di una funzione su tutti gli elementi di un dizionario indicato.
La prima operazione sviluppata durante il workflow è stata la definizione del type exp, in cui ho definito il nuovo tipo "Edict". Questo nuovo tipo è composto da un ulteriore nuovo tipo "dict" così definito:

```
dict := Empty | Item of (ide * exp) * dict
```

Quindi il dizionario può essere composto da un elemento vuoto o da un item composto da ide * exp e, ricorsivamente, da un dict.
Successivamente sono state implementate l'operazione di Clear, ApplyOver, Get, Rm e Add.

- Clear prende una sola exp che si assume sia un Edict;
- ApplyOver prende due exp: la prima si assume sia una Fun e un Edict;
- Get prende un ide, che corrisponde all'identificatore da ricercare nel dizionario, e un exp, ossia il dizionario Edict in cui bisogna ricercare il valore corrispondente all'identificatore;
- Rm prende un ide, che corrisponde all'identificatore da eliminare dal dizionario e un exp, ossia il dizionario Edict da cui bisogna eliminare l'Item corrispondente all'identificatore;
- Add prende una coppia (ide * exp) che corrisponde a un singolo Item da aggiungere nel dizionario Edict, identificato dall'elemento exp di Add.

Successivamente è stato implementato il tipo dei valori esprimibili (evT) in cui, oltre a tutti i valori esprimibili di default, è stato aggiunto anche il valore esprimibile `DictVal of evDic` in cui evDic ha la seguente struttura grammaticale:

```
type evDic = EvEmpty | Elem of (ide * evT) * evDic
```

in cui EvEmpty è il tipo esprimibile del dizionario vuoto ed Elem of (ide * evT) * evDic è la definizione del valore esprimibile per l'elemento del dizionario.

Dopo aver implementato le funzioni per definire il typecheck e le operazioni standard e le varie operazioni nell'eval corrispondenti alle operazioni standard sono stati implementati i case del "match e with" dell'eval per le operazioni su dizionario:

- Edict restituisce un valore esprimibile del dizionario per mezzo della funzione ausiliaria `eval_d` che, preso un dizionario e un ambiente, restituisce il valore esprimibile del dato dizionario nell'ambiente dato.
- Clear restituisce semplicemente un DictVal (definito sopra) corrispondente al dizionario vuoto
- ApplyOver restituisce il valore esprimibile del risultato della funzione `apply_o`, questa funzione è stata realizzata rifacendomi all'Apply dell'eval: matcho il valore esprimibile della funzione con i vari casi: Closure e RecClosure, contemporaneamente matcho il valore esprimibile del dizionario. Se la funzione non è ricorsiva allora applico la funzione sul singolo Item del dizionario e richiamo ricorsivamente apply_o sul resto del dizionario, se, invece, è ricorsiva, creo un ambiente in cui faccio il bind tra la funzione g (il nome della funzione

ricorsiva) ed f (il nome della funzione della funzione che sto usando) , poi creo un nuovo ambiente in cui bindo l'argomento della funzione ricorsiva a el (il primo elemento del dizionario) e, successivamente, restituisco il valore esprimibile Elem in cui l'identificatore rimane invariato, l'elemento corrispondente all'identificatore viene valutato nel corpo della funzione usando l'ambiente aEnv, quello creato per bindare l'argomento della funzione ricorsiva all'elemento del dizionario, e poi applico la funzione ricorsivamente al resto del dizionario.

- Get usa una funzione `search_d` per ricercare ricorsivamente (se esiste) , dato un valore esprimibile di dizionario e un ide, l'elemento corrispondente all'ide dato, restutuendo `true`, altrimenti restituisce `false`.
- Rm restituisce un dizionario da cui è stato rimosso l'elemento per mezzo della funzione remove_d la quale, dato il valore esprimibile del dizionario, l'identificatore e l'ambiente, cerca l'elemento e, se lo trova, restituisce il resto del dizionario (in valore esprimibile), altrimenti continua la ricerca.
- Add aggiunge l'elemento in coda al dizionario e resistuisce il dizionario corrispondente, utilizza la funzione Add_d che, dato un dizionario, un identificatore, un valore esprimibile (che corrisponderà all'identificatore) e un ambiente, scorre tutto il dizionario e, quando arriva alla fine, aggiunge l'elemento (i * eEl).

Le operazioni su dizionario sono state implementate restituendo valori esprimibili al fine di essere valutate in altre operazioni.

Riferimenti: (Link diretto al codice su GitHub) https://github.com/fulviodenza/Ocaml-Interpreter