Relazione Interprete esteso con tipi di dato *Dizionario*

Ho utilizzato la sintassi astratta vista a lezione, estendendola leggermente per poter utilizzare il tipo di dato *Dict*, che implementa un dizionario di elementi. Oltre alle operazioni su interi e booleani già viste, è infatti presente il costruttore

```
|Dict of edic
:
and edic = Empty | Elem of (ide * exp) * edic
```

L'idea iniziale era utilizzare una lista di coppie (*ide * exp*), rendendo l'oggetto più vicino all'idea di Dizionario che ha l'utente ed il codice più elegante e leggibile.
Questa è stata abbandonata, in quanto avrebbe diminuito l'astrazione della sintassi, costringendo ad utilizzare un meccanismo concreto del linguaggio.
Ho definito un costruttore per espressioni di tipo stringa (*Estring of string*), adattando opportunamente i tipi di dato esprimibili *evT* e, qualora si volessero effettuare operazioni su dati di tipo string, il *run time support* per il controllo di tipi.
Analogamente ho definito i costruttori per le operazioni su dati di tipo dizionario:

```
type exp =
:
| DictAccess of evT * ide
| AddInDict of evT * ide * exp
| RmDict of evT * ide
| Clr of evT
| ApplyOver of exp * evT
```

Ognuno di questi costrutti lavora su un dizionario già valutato *evT*.
DictAccess prende come parametro la chiave dell'oggetto al quale si vuole accedere, restituendo un valore *ElemDict(v)*, dove v è il valore associato alla chiave.*AddInDict* prende come parametro un dizionario ed una coppia (k,v), e si occupa di aggiungere al primo la coppia (*k, eval v r*).
RmDict rimuove l'oggetto che corrisponde alla chiave passata, se questo è nella collezione.
Clr effettua l'operazione *Clear*, restituendo un dizionario completamente vuoto.
ApplyOver, intuitivamente, prende una funzione e la applica al dizionario passato.
Ho utilizzato una funzione *toexp(a : evT)* che permette di convertire un oggetto *evT* trasportante un intero, un booleano o una stringa in un'analoga espressione *exp*. Questo meccanismo è di supporto all'interprete durante l'esecuzione dell'ApplyOver a tutti gli elementi del dizionario.
Per quanto riguarda la parte opzionale del progetto, la scelta che ho fatto è di creare un nuovo interprete che esegua le operazioni con regole di scope dinamico. Le modifiche necessarie per tale meccanismo, infatti, sono poche e consistono nel valutare la funzione nell'ambiente globale r anziché in quello di definizione e l'estensione del tipo di dato *evT*, inserendo due nuovi costrutti per la valutazione delle funzioni (ricorsive e non) con regole di scope dinamico. Questi costrutti, a differenza della definizione prevista per la chiusura di una funzione, non ne comprendono l'ambiente di definizione.
I test presenti nel file riguardano ogni tipo di operazione, su dati leciti e non (Ci si aspetta un'interruzione dell'esecuzione, motivo per cui questi test sono commentati), e sulla valutazione di una funzione con regole di scope statico e dinamico.

Marco Antonio Corallo, 531466