Relazione

Secondo progetto intermedio - A.A. 2020-2021

Luca De Paulis

1 ESEGUIRE IL PROGETTO

Per eseguire il progetto nella REPL standard `ocaml` o in `utop` bisogna invece innanzitutto compilare i file `OCaml` tramite il comando `ocamlc` nel seguente modo:

```
$ ocamlc interpreter.ml
$ ocamlc interpreter.cmo typechecker.ml
$ ocamlc interpreter.cmo typechecker.cmo tests.ml
```

A questo punto si possono mandare in esecuzione i vari test (o valutare altre espressioni) caricando i moduli nella REPL:

```
#load "interpreter.cmo";;
#load "typechecker.cmo";;
#use "tests.ml";;
```

2 STRUTTURA DELL'INTERPRETE

L'interprete `eval` lavora sull'albero di sintassi astratta dato dal tipo `expr` e, dato un ambiente con tipo `evT env`, restituisce un valore esprimibile `evT` oppure un errore. Similmente il typechecker statico `typeof` prende un'espressione di tipo `expr` e un ambiente dei tipi con tipo `typ env` e restituisce un errore oppure un tipo del linguaggio (`typ`).

Entrambe le funzioni hanno una versione che usa come ambiente di default l'ambiente vuoto, chiamate rispettivamente `eval'` e `typeof'`. Inoltre viene definita la funzione `check_eval` che prende un'espressione, ne calcola il tipo nell'ambiente vuoto e se non vi sono errori restituisce la valutazione dell'espressione nell'ambiente vuoto.

Estensioni all'interprete didattico

Rispetto alla versione studiata a lezione, le espressioni `Fun` e `LetRec` necessitano anche del tipo del parametro e del tipo di ritorno della funzione. Questi tipi vengono controllati dal typechecker statico e in caso non combacino con i tipi effettivi viene sollevata un'eccezione.

Inoltre il tipo del parametro e il tipo di ritorno vengono inclusi nelle chiusure e nelle chiusure ricorsive: in questo modo vengono mantenute delle informazioni utili al typechecker dinamico per la valutazione degli operatori funzionali sugli insiemi.

Estensione agli insiemi

Un insieme è un valore esprimibile caratterizzato da un tipo (il tipo degli elementi) e da una lista di valori costruita con un il tipo di dato algebrico `set_val`.

Un insieme può essere costruito tramite le espressioni **`EmptySet`**, **`Singleton`** e **`Of`**. La lista di espressioni da passare a **`Of`** (di tipo `set_expr`) può essere creata con i costruttori **`Empty`** e **`Cons`**, oppure con l'operatore `( @: )` che è semplicemente zucchero sintattico per **`Cons`**.

Il typechecker statico si assicura che le espressioni passate a **`Singleton`** e a **`Of`** siano del tipo corretto; l'interprete fa lo stesso controllo con un typechecker dinamico ed inoltre fa in modo che ogni valore contenuto nell'insieme compaia al più una volta.

Nel caso delle operazioni di base sugli insiemi (come ad esempio **`Union`** oppure **`Inters`**) viene svolto sempre un controllo (statico nel caso della `typeof`, dinamico nel caso della `eval`) sui tipi dei parametri; inoltre viene sempre mantenuto l'invariante dell'assenza di ripetizioni di elementi. Le espressioni **`MinOf`** e **`MaxOf`** sollevano un'eccezione nel caso in cui l'insieme sia vuoto.

Gli operatori funzionali sfruttano le informazioni sul tipo della funzione contenute nella chiusura per controllare dinamicamente che i tipi delle funzioni siano corretti e, nel caso della **`Map`**, anche per determinare il tipo dell'insieme restituito.

Semantica operazionale

Le regole della semantica operazionale sono definite nel file `semantics.pdf`. Ogni espressione valutabile viene descritta da una regola principale, data in semantica naturale, che rappresenta come calcolare il valore a partire dall'espressione in sintassi astratta, e da alcune regole ausiliarie che formalizzano il calcolo delle operazioni di base (come ad esempio l'unione insiemistica) sui valori di tipo `set_val`.