Progetto interprete Scheme (parte opzionale)

Roberto Lucchi

Linguaggi di Programmazione 2005/06

Consegna progetti

- Una data di consegna per appello
  - La prova orale si può effettuare solo se il progetto viene discusso nella stessa sessione
  - Data consegna prossima sessione: 26 febbrario 2006
  - La data della sessione estiva verrà comunicata

Sistema di tipi

- Abbiamo la parte di tipi di dato in minicheme che è definita nel seguente frammento:

$$\begin{array}{lcl} \langle const \rangle & ::= & \texttt{\#t} \mid \texttt{\#f} \\ & \mid & int \\ & \mid & string \end{array}$$

- Anche *id* (identificativo di funzione) è un tipo

Linguaggi tipati, quali controlli?

- (and espr1 espr2)
  - espr1 e espr2 devono essere espressioni booleane (cioè si riducono ad un boolean)
  - Casi base: #t e #f sono espr booleane
- Lo stesso per (or espr1 espr2)

Progetto type checker

- Dovete definire ed implementare una serie di regole che definiscono le espressioni ben tipate (ovviamente la valutazione di espressioni può coinvolgere funzioni) considerando
  - le espressioni <expr> definite (grammatica)
  - le operazioni primitive, ad esempio
    - (+ espr1 espr2)
      - espr1 e espr2 devono essere ridotte a interi (tipo *int*)
      - Casi base: ogni intero è di tipo *int*
    - cons che definisce il tipo *pair*
    - car l (prima componente della coppia l) si applica solo se l è una coppia

Progetto type checker

- Questa parte deve essere integrata con la parte **base** del progetto (interprete)

Idea

- Associare ad ogni *valore* (booleano, intero, coppia, chiusura) ed *espressione* (metodo evaluate che restituisce un valore) un *tipo*
- Nota:
  - In minischeme non esiste la dichiarazione esplicita del tipo di valore/nome, il loro tipo è determinato ***dinamicamente***.

    (define (somma a b) (cond((= a b)+ a b) …))

    (somma "mario" 23) o (somma #t 2)

    a e b della funzione somma possono contenere qualsiasi valore, anche stringa, ma ovviamente l'operatore + può essere applicato solo fra interi (così come = può essere applicato solo fra interi)

Progetto

- Potete anche pensare di estendere la parte di interprete per esplicitare meglio condizioni di errore.
- Esempio:

$$[\textsc{Cond3}]\ \frac{T_i \to \texttt{\#f}, \forall i \in \{1,\ldots,n\} \quad E_{n+1} \to v}{\begin{array}{c}(\texttt{cond}\ (T_1\ E_1)\ \cdots\ (T_n\ E_n) \\ (\texttt{else}\ E_{n+1})) \to v\end{array}}$$

  la regola non considera che le guardie T_i debbano essere espressioni booleane

Progetto

- Definire un sistema di type inference per minischeme considerando i tipi base supportati, le funzioni primitive offerte e i costrutti di base
  - Commentato nella documantazione del progetto
- Estendere le strutture dati del progetto base, a partire dalle interfacce date, in modo tale da implementare un type cheker dinamico, che applichi le regole di type inference, in grado di verificare che il programma sia *ben formato* (suggerimento, partire da SchemeValue e FactoryImpl che implementa SchemeFactory)
  - Nel caso di errore quindi deve essere in grado di segnalarlo durante l'esecuzione

Libro di riferimento

- Compilers, Principles, Techniques and Tools

  A. V. Aho, R. Sethi, J. D. Ullman
  - Essenzialmente il capitolo 6