CuCh machine

Linguaggi di Programmazione
a.a. 2019-2020

Edoardo De Matteis 1746561
Mirko Giacchini 1811809

Indice

1 Introduzione

Una **Cu**rry-**Ch**urch machine implementa un interprete minimale di un linguaggio funzionale non tipato. Abbiamo quindi implementato in SML un interprete per il linguaggio FUN visto a lezione. Nonostante una CuCh machine dovrebbe implementare l'applicazione di funzioni per sostituzione, si è qui preferito utilizzare una valutazione basata su ambienti, questo per poter sperimentare anche con semantiche differenti. La valutazione per sostituzione può comunque essere emulata con la semantica lazy statica.
È stato poi usato l'interprete creato per scrivere alcuni programmi di esempio (tra cui booleani di Church, numeri di Church, ecc.).

2 Sintassi

syntax.sml

$$\begin{aligned} FUN ::= & \; Const\ k \mid Var\ x \\ & \mid Sum(M, N) \mid Fn(x, M) \\ & \mid Let(x, M, N) \mid App(M, N) \end{aligned} \tag{1}$$

$$K ::= 0 \mid 1 \mid \cdots \tag{2}$$

$$X ::= A \mid \cdots \mid Z \mid a \mid \cdots \mid z \mid \cdots \tag{3}$$

$$ENV : VAR \rightharpoonup VAR \times FUN \times ENV \tag{4}$$

$$find : ENV \times VAR \to (FUN \times ENV) \cup EXC \tag{5}$$

In ENV nel codominio il prodotto cartesiano presenta ENV poichè necessario nelle valutazioni con scoping statico, nel mondo dinamico non è necessario e semplicemente lo si ignora. EXC è l'insieme delle eccezioni.

3 Semantica operazionale

$$\mapsto \subseteq ENV \times FUN \times VAL \;\equiv\; ENV \vdash FUN \mapsto VAL \tag{6}$$

3.1 Dynamic eager

dynamic_ eager.sml

$$\frac{}{E \vdash Const\ k \mapsto Const\ k}$$

$$\frac{}{E \vdash Var\ x \mapsto v} \quad \mathrm{E(x)} = \mathrm{v}$$

$$\frac{E \vdash M \mapsto v1 \qquad E \vdash N \mapsto v2}{E \vdash Sum(M,N) \mapsto v} \quad (\mathrm{v}{=}\mathrm{v1}{+}\mathrm{v2})$$

$$\frac{}{E \vdash Fn(x,M) \mapsto (x,M)}$$

$$\frac{E \vdash M \mapsto (x,M') \qquad E \vdash N \mapsto v \qquad E(x,v) \vdash M' \mapsto v'}{E \vdash App(M,N) \mapsto v'}$$

$$\frac{E \vdash M \mapsto v \qquad E(x,v) \vdash N \mapsto v'}{E \vdash Let(x,M,N) \mapsto v'}$$

3.2 Dynamic lazy

dynamic_ lazy.sml

$$\frac{}{E \vdash Const\ k \mapsto Const\ k}$$

$$\frac{E \vdash M \mapsto v}{E \vdash Var\ \ x \mapsto v} \quad \mathrm{E(x) = M}$$

$$\frac{E \vdash M \mapsto v1 \qquad E \vdash N \mapsto v2}{E \vdash Sum(M, N) \mapsto v} \quad \mathrm{(v{=}v1{+}v2)}$$

$$\frac{}{E \vdash Fn(x, M) \mapsto (x, M)}$$

$$\frac{E \vdash M \mapsto (x, M') \qquad E(x, N) \vdash M' \mapsto v}{E \vdash App(M, N) \mapsto v}$$

$$\frac{E(x, M) \vdash N \mapsto v}{E \vdash Let(x, M, N) \mapsto v}$$

3.3 Static eager

static_ eager.sml

$$\frac{}{E \vdash Const\ k \mapsto Const\ k}$$

$$\frac{}{E \vdash Var\ x \mapsto v} \quad \mathrm{E(x) = v}$$

$$\frac{E \vdash M \mapsto v1 \qquad E \vdash N \mapsto v2}{E \vdash Sum(M, N) \mapsto v} \quad \mathrm{(v{=}v1{+}v2)}$$

$$\frac{}{E \vdash Fn(x, M) \mapsto (x, M, E)}$$

$$\frac{E \vdash M \mapsto (x, M', E') \qquad E \vdash N \mapsto v \qquad E'(x, v) \vdash M' \mapsto v'}{E \vdash App(M, N) \mapsto v'}$$

$$\frac{E \vdash M \mapsto v \qquad E(x, v) \vdash N \mapsto v'}{E \vdash Let(x, M, N) \mapsto v'}$$

3.4 Static lazy

static_ lazy.sml

$$\frac{}{E \vdash Const\ k \mapsto Const\ k}$$

$$\frac{E' \vdash M \mapsto v}{E \vdash Var\ \ x \mapsto v} \quad \text{E(x) = (M, E')}$$

$$\frac{E \vdash M \mapsto v1 \qquad E \vdash N \mapsto v2}{E \vdash Sum(M, N) \mapsto v} \quad \text{(v=v1+v2)}$$

$$\frac{}{E \vdash Fn(x, M) \mapsto (x, M, E)}$$

$$\frac{E \vdash M \mapsto (x, M', E') \qquad E'(x, N) \vdash M' \mapsto v}{E \vdash App(M, N) \mapsto v}$$

$$\frac{E(x, M, E) \vdash N \mapsto v}{E \vdash Let(x, M, N) \mapsto v}$$

4 Esempi

church_ bool.sml
Sono stati implementati i booleani di Church, interamente in linguaggio FUN, e qualche semplice operazione sui booleani.

church_ numbers.sml
Sono stati implementati i numeri di Church, interamente in linguaggio FUN, e qualche operazione tra numeri di Church (somma, prodotto, potenza).

church_ factorial.sml
È stato implementato il fattoriale usando solamente i numeri di Church (quindi è stata necessaria l'implementazione di una funzione isZero e di una funzione predecessore). Essendo questo programma più complesso, abbiamo usato il nostro interprete "immerso" in SML, in modo da poter usare variabili SML come degli alias, in modo da renderlo più leggibile.

church_ bintree.sml
Sono stati implementati gli alberi binari di Church, con una funzione che ne conta le foglie. Notiamo che gli alberi binari di Church non sono implementabili in SML a causa del sistema dei tipi, in FUN non tipato riusciamo invece a creare un esempio funzionante e la sua realizzazione è stata lo scopo principale di questo progetto.

5 Osservazioni[1]

Un programma che mette in mostra le differenze tra una valutazione eager ed una lazy è il programma

$$let\ x = x\ in\ x$$

che in una semantica static eager (come in SML) dà errore perchè x non è definita. In una semantica dynamic lazy va invece in loop

$$\dfrac{\dfrac{\dfrac{\dfrac{\dots}{\langle (x,x)\rangle \vdash x \mapsto}}{\langle (x,x)\rangle \vdash x \mapsto}}{\emptyset \vdash let\ x = x\ in\ x \mapsto}}$$

Inoltre in SML non è possibile eseguire il più piccolo transinfinito

$$\omega = (fn\ x \Rightarrow x\ x)(fn\ x \Rightarrow x\ x)$$

per via del sistema dei tipi, FUN non essendo tipato non presenta questo problema e si entra in loop come si può dimostrare ad esempio con valutazione eager statica

$$\dfrac{\emptyset \vdash fn\ x \Rightarrow x\ x \mapsto (x, x\ x, \emptyset) \qquad \emptyset \vdash fn\ x \Rightarrow x\ x \mapsto (x, x\ x, \emptyset) \qquad \dfrac{\dfrac{\dfrac{\dots}{\langle (x, x\ x)\rangle \vdash x\ x \mapsto}}{\langle (x, x\ x)\rangle \vdash x\ x \mapsto}}{\langle (x, x\ x)\rangle \vdash x\ x \mapsto}}{\emptyset \vdash (fn\ x \Rightarrow x\ x)(fn\ x \Rightarrow x\ x) \mapsto}$$

Un'esecuzione interessante è quella di $\langle (x,k)\rangle \vdash (fn\ x \Rightarrow x)x$ che in ambiente lazy dinamico ha un comportamento per cui la valutazione cambia in base a come si rinominano le variabili violando l'α-regola.

$$\dfrac{\langle (x,3)\rangle \vdash fn\ x \Rightarrow x \mapsto (x,x) \qquad \dfrac{\dfrac{\dots}{\langle (x,3)(x,x)\rangle \vdash x \mapsto}}{\langle (x,3)(x,x)\rangle \vdash x \mapsto}}{\langle (x,3)\rangle \vdash (fn\ x \Rightarrow x)x \mapsto}$$

$$\dfrac{\langle (x,3)\rangle \vdash fn\ y \Rightarrow y \mapsto (y,y) \qquad \langle (x,3)(y,y)\rangle \vdash x \mapsto 3}{\langle (x,3)\rangle \vdash (fn\ y \Rightarrow y)x \mapsto 3}$$

È interessante notare che nell'implementazione delle funzioni fattoriale e numLeaves (degli esempi precedenti) è stato necessario l'uso del ricorsore (7). Eseguendo tali programmi con semantiche diverse dalla lazy statica (cioè quella

[1] Negli esempi proposti si è preferito adottare la sintassi di FUN per leggibilità.

che corrisponde a una valutazione per sostituzione) non si ottiene il risultato voluto in quanto si va loop.

$$\mathrm{Y} = fn\ f \Rightarrow (fn\ x \Rightarrow f(xx))(fn\ x \Rightarrow f(xx)) \tag{7}$$