Compitino su Semantica Operazionale

Si consideri la seguente variante di HOFL

$$\begin{array}{rcl} l \in \mathsf{Loc} & ::= & \text{locazioni, p.e. i numeri naturali} \\ n \in \mathsf{N} & ::= & \text{i numeri interi} \\ x \in \mathsf{Var} & ::= & \ldots \\ t \in \mathsf{Exp} & ::= & n \mid t_1 + t_2 \mid \mathsf{if}\ t_0\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \mid x \mid \lambda x.t \mid t_1\ t_2 \mid \mathsf{rec}\ x.t \mid l \mid t_1 := t_2 \mid !t \mid \mathsf{new}\ x := t_1\ \mathsf{in}\ t_2 \\ v \in \mathsf{Val} & ::= & n \mid \lambda x.t \mid l \end{array}$$

dove i nuovi costrutti hanno la seguente semantica informale

- $t_1 := t_2$, se t_1 e' una locazione gia' esistente l e t_2 ha valore n, allora assegna n ad l e restituisci l
- $!t$, se t e' una locazione gia' esistente l, allora restituisci il contenuto n di l
- $\mathsf{new}\ x := t_1\ \mathsf{in}\ t_2$, se t_1 ha valore n, allora rimpiazza x con una nuova locazione l con contenuto n, e poi valuta t_2

1. Definire l'insieme $FV(t) \subseteq_{fin} \mathsf{Var}$ delle variabili libere e $UL(t) \subseteq_{fin} \mathsf{Loc}$ delle locazioni usate in t. **Per evitare confusione tra interi e locazioni, scriveremo l_n per indicare la locazione corrispondente al numero naturale n.** Usare una tabella

$t \in \mathsf{Exp}$	$FV(t)$	$UL(t)$
l	$\emptyset$	$\{l\}$
...	...	...

2. Dare due semantiche operazionali CBV a grandi passi, con asserzioni della forma $t, \mu \to v, \mu'$ dove $t \in \mathsf{Exp}_0$, $v \in \mathsf{Val}_0$ e $\mu, \mu' \in \mathsf{S} \triangleq \{(n, s) | n \in \mathsf{Nat} \wedge s : n \to \mathsf{N}\}$. In altre parole $\mu = (n, s)$ e' uno stato con n locazioni $l \in n = \{0, 1, \ldots, n-1\}$ (cioe' identifichiamo un numero naturale con l'insieme dei suoi predecessori), e $s : n \to \mathsf{N}$ assegna a ciascuna di queste n locazioni un numero intero.

   Una semantica deve formalizzare la gestione della memoria a stack (come in C e Pascal), mentre l'altra quella ad heap (come in CAML). **L'unica regola che dovrebbe distinguere tra le due semantiche e' quella per la valutazione di $\mathsf{new}\ x := t_1\ \mathsf{in}\ t_2$.**

3. Dimostrare che nel caso di gestione a stack si ha che $t, (n, s) \to v, (n', s') \Longrightarrow n = n'$, mentre nel caso di gestione a heap si ha che $t, (n, s) \to v, (n', s') \Longrightarrow n \le n'$.

   considerare solo i casi che riguardano new, ed eventualmente l'applicazione

4. Dimostrare che nel caso di gestione a heap si ha che $t, (n, s) \to v, (n', s') \wedge UL(t) \subseteq n \Longrightarrow UL(v) \subseteq n'$. Questa proprieta' non vale nel caso di gestione a stack, p.e.

   $$\mathsf{new}\ x := 0\ \mathsf{in}\ x, (0, \emptyset) \to l_0, (0, \emptyset)$$

   considerare solo i casi che riguardano new, ed eventualmente l'applicazione

   [DIFFICILE/FACOLTATIVO] dire come restringere la regola di valutazione di new-in, in modo da conservare la gestione a stack, ma evitare che si usino locazioni non presenti nello stato.

Soluzione

1.

$t \in \mathsf{Exp}$	$FV(t)$	$UL(t)$
n	$\emptyset$	$\emptyset$
$t_1 + t_2$	$FV(t_1) \cup FV(t_2)$	$UL(t_1) \cup UL(t_2)$
$\mathsf{if}\ t_0\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2$	$FV(t_0) \cup FV(t_1) \cup FV(t_2)$	$UL(t_0) \cup UL(t_1) \cup UL(t_2)$
x	$\{x\}$	$\emptyset$
$\lambda x.t$	$FV(t) - \{x\}$	$UL(t)$
$t_1\ t_2$	$FV(t_1) \cup FV(t_2)$	$UL(t_1) \cup UL(t_2)$
$\mathsf{rec}\ x.t$	$FV(t) - \{x\}$	$UL(t)$
l	$\emptyset$	$\{l\}$
$t_1 := t_2$	$FV(t_1) \cup FV(t_2)$	$UL(t_1) \cup UL(t_2)$
$!t$	$FV(t)$	$UL(t)$
$\mathsf{new}\ x := t_1\ \mathsf{in}\ t_2$	$FV(t_1) \cup (FV(t_2) - \{x\})$	$UL(t_1) \cup UL(t_2)$

2. $$\frac{}{n,\mu_0 \to n,\mu_0} \qquad \frac{t_1,\mu_0 \to n_1,\mu_1 \qquad t_2,\mu_1 \to n_2,\mu_2}{t_1+t_2,\mu_0 \to n,\mu_2}\ n = n_1+n_2$$

$$\frac{t_0,\mu_0 \to 0,\mu_1 \qquad t_1,\mu_1 \to v,\mu_2}{\mathsf{if}\ t_0\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2,\mu_0 \to v,\mu_2} \qquad \frac{t_0,\mu_0 \to n,\mu_1 \qquad t_2,\mu_1 \to v,\mu_2}{\mathsf{if}\ t_0\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2,\mu_0 \to v,\mu_2}\ n \neq 0$$

$$\frac{}{\lambda x.t,\mu_0 \to \lambda x.t,\mu_0} \qquad \frac{t_1,\mu_0 \to \lambda x.t,\mu_1 \qquad t_2,\mu_1 \to v,\mu_2 \qquad t[x:=v],\mu_2 \to v',\mu_3}{t_1\ t_2,\mu_0 \to v',\mu_3}$$

$$\frac{t[x:=\mathsf{rec}\ x.t],\mu_0 \to v,\mu_1}{\mathsf{rec}\ x.t,\mu_0 \to v,\mu_1} \qquad \frac{}{l,\mu_0 \to l,\mu_0} \qquad \frac{t,\mu_0 \to l_m,(n_1,s_1)}{!t,\mu_0 \to n,(n_1,s_1)}\ n = s_1(l_m)$$

$$\text{set*}\ \frac{t_1,\mu_0 \to l_m,(n_1,s_1) \qquad t_2,(n_1,s_1) \to n,(n_2,s_2)}{t_1:=t_2,\mu_0 \to l_m,(n_2,s_2[l_m:=n])}\ l_m \in n_1 \text{ e } n_1 \leq n_2$$

$$\text{heap}\ \frac{t_1,\mu_0 \to n,(n_1,s_1) \qquad t_2[x:=l_{n_1}],(n_1+1,s_1[l_{n_1}:=n]) \to v,\mu_2}{\mathsf{new}\ x:=t_1\ \mathsf{in}\ t_2,\mu_0 \to v,\mu_2}$$

$$\text{stack*}\ \frac{t_1,\mu_0 \to n,(n_1,s_1) \qquad t_2[x:=l_{n_1}],(n_1+1,s_1[l_{n_1}:=n]) \to v,(n_2,s_2)}{\mathsf{new}\ x:=t_1\ \mathsf{in}\ t_2,\mu_0 \to v,(n_1,s_2\lceil n_1)}\ n_1 \leq n_2$$

$s\lceil n$ indica la funzione s ristretta al dominio n, cioe' $\lambda x \in n.s(x)$.

La side-condition $n_1 \leq n_2$ in (set*) e (stack*). si puo' dimostrare a posteriori che puo' essere eliminata.

3. $t,(n,s) \to v,(n',s') \Longrightarrow n = n'$ (nella gestione a stack) si dimostra per induzione sulla derivazione di $t,(n,s) \to v,(n',s')$. In tutti i casi si applica in modo ovvio l'ipotesi induttiva. Nel caso della regola (stack*) si ha:

   - $n_1 = n_0$ per IH applicata alla derivazione della prima premessa [e convenianmo che $(n_0,s_0) \equiv \mu_0$], questo e' gia' sufficiente per concludere la dimostrazione
   - $n_2 = n_1+1$ per IH applicata alla derivazione della seconda premessa [quindi la condizione $n_1 \leq n_2$ e' inutile]

   $t,(n,s) \to v,(n',s') \Longrightarrow n \leq n'$ (nella gestione a heap) si dimostra per induzione sulla derivazione di $t,(n,s) \to v,(n',s')$. In tutti i casi si applica in modo ovvio l'ipotesi induttiva. Nel caso della regola (heap*) si ha:

   - $n_1 \geq n_0$ per IH applicata alla derivazione della prima premessa
   - $n_2 \geq n_1+1$ per IH applicata alla derivazione della seconda premessa, questo e' sufficiente per concludere che $n_2 \geq n_0$ [in effetti in questo caso si ha $n_2 > n_0$].

4. $t,(n,s) \to v,(n',s') \wedge UL(t) \subseteq n \Longrightarrow UL(v) \subseteq n'$ (nella gestione ad heap) si dimostra per induzione sulla derivazione di $t,(n,s) \to v,(n',s')$.

   In tutti i casi si applica in modo ovvio l'ipotesi induttiva, ed la proprieta' della semantica operazionale dimostrata al punto precedente. Tuttavia nelle regole dove si effettuano delle sostituzioni, si deve sfruttare anche la seguente **proprieta' sostitutiva di UL** $UL(t_2[x:=t_1]) \subseteq UL(t_1) \cup UL(t_1)$.

   Nel caso della regola (heap*) si deve dimostrare che $UL(v) \subseteq n_2$ sapendo che $UL(\mathsf{new}\ x:=t_1\ \mathsf{in}\ t_2) \subseteq n_0$:

   - $n_1 \geq n_0$ per il punto precedente, e quindi $UL(t_2) \subseteq UL(\mathsf{new}\ x:=t_1\ \mathsf{in}\ t_2) \subseteq n_1$
   - $UL(t_2[x:=l_{n_1}]) \subseteq UL(t_2) \cup \{l_{n_1}\} \subseteq n_1+1$ per la proprieta' sostitutiva di UL
   - $UL(v) \subseteq n_2$ per IH applicata alla derivazione della seconda premessa.

5. PARTE FACOLTATIVA.

   Si puo' modificare la regola (stack*) richiedendo che il corpo del new restituisca un intero n, invece che un valore qualunque v, infatti $UL(n) = \emptyset \subseteq n_1$.

$$\text{stack**}\ \frac{t_1,\mu_0 \to n,(n_1,s_1) \qquad t_2[x:=l_{n_1}],(n_1+1,s_1[l_{n_1}:=n]) \to n,(n_2,s_2)}{\mathsf{new}\ x:=t_1\ \mathsf{in}\ t_2,\mu_0 \to n,(n_1,s_2\lceil n_1)}\ n_1 \leq n_2$$