Models of Computation

Written Exam on January 8, 2013

(MOD students: Exercises 1-5, 180 minutes   Previous TSD students: Exercises 1-3, 120 minutes)

Exercise 1 (8)

Given the IMP command

$$\mathrm{c} = \textbf{while } x \neq 0 \textbf{ do } (x := x - 1 \; ; \; y := y + 1 \; ; \; z =: z \times y)$$

prove that, for every $\sigma, \sigma' \in \Sigma$,

$$\langle c, \sigma \rangle \rightarrow \sigma' \text{ implies } \sigma(y) > 0 \Rightarrow \sigma(x) \geq 0 \wedge \sigma' = \sigma[0/x, \sigma(y){+}\sigma(x)/y, n/z]$$
$$\text{with } n = \sigma(z)(\sigma(y){+}\sigma(x))!/\sigma(y)!$$

while $\sigma(x) < 0 \Rightarrow \langle c, \sigma \rangle \not\rightarrow$.

Exercise 2 (6)

Given a signature Σ, consider the relation $\sqsubseteq = <^*$ between terms $t \in T_\Sigma$ in Σ which is the reflexive and transitive closure of the subterm-term relation $<$ defined as $t_i < f(t_1, \ldots, t_n),\ 1 \leq i \leq n$. Prove that $\sqsubseteq$ is a partial ordering. Finally discuss when, depending on particular choices of Σ, such an ordering is complete and/or with bottom.

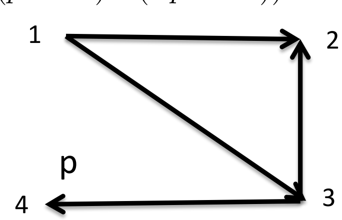
Exercise 3 (6)

Extend the syntax of HOFL lazy adding the *pairing* construct $< t_1, t_2 >$ of functions $t_1 : \sigma \longrightarrow \tau_1$ and $t_2 : \sigma \longrightarrow \tau_2$, both having arguments of type σ. Here $< t_1, t_2 >: \sigma \longrightarrow \tau_1 * \tau_2$ defines the function which, given an argument d, returns the pair (r_1, r_2), where r_1 is the result of t_1 given d e r_2 the result of t_2 given d.

For the new construct define: (i) a type rule; (ii) the operational semantics, in such a way that the construct is immediately reduced to canonical form; and (iii) the denotational semantics. Finally prove that, for every (closed) term (t), the terms $(< t_1, t_2 > \ t)$ and $((t_1 \ t), (t_2 \ t))$ have: (i) the same type; (ii) the same canonical form; and (iii) the same denotational semantics.

Exercise 4 (5)

Given the μ-calculus formula $\Phi = \mu x.((p \wedge \Box x) \vee (\neg p \wedge \Diamond x))$ write its denotational semantics $[\![\Phi]\!]\rho$ and evaluate it on the LTS below.



Exercise 5 (5)

Consider the π agent $P = (x)(P_1|P_2)$, with $P_1 = !((y)\overline{z}y.!(\overline{x}y.nil))$ and $P_2 = !(x(y).x(y').\overline{y}y'.nil)$, and describe informally its behavior. In particular, define a sequence of transitions from P (which has just the free name z) to a state P' where there are three additional free names y_1, y_2, y_3, and the behavior of P' includes any number of transitions labelled by $\overline{y_i}y_j$ for $y_i, y_j = y_1, y_2, y_3$.

Prova Scritta del 08/01/13

Esercizio 1

a) Per induzione sulle regole del while.

$$P(\langle c,\sigma\rangle \to \sigma') \overset{def}{=} \sigma(y)>0 \Rightarrow \sigma(x)\geq 0 \wedge \sigma' = \sigma[0/x, \sigma(x)+\sigma(y)/y, n/z]$$

$$n = \sigma(z)(\sigma(y)+\sigma(x))!/\sigma(y)!$$

$$\frac{\sigma(x)=0}{\langle c,\sigma\rangle \to \sigma} \quad P(\langle c,\sigma\rangle \to \sigma) = \sigma(y)>0 \overset{?}{\Rightarrow} \sigma(x)\geq 0 \wedge \sigma = \sigma[0/x, \sigma(y)/y, \sigma(z)/z]$$

ovvio. CVD.

$$\frac{\sigma(x)\neq 0 \quad \langle x:=x-1; y:=y+1; z:=z\times y, \sigma\rangle \to \sigma'' \quad \langle c,\sigma''\rangle \to \sigma'}{\langle c,\sigma\rangle \to \sigma'}$$

$$\sigma(x)\neq 0 \quad \sigma'' = \sigma[\sigma(x)-1/x, \sigma(y)+1/y, \sigma(z)(\sigma(y)+1)/z]$$

Assumiamo le premesse $\sigma(y)>0$. Quindi anche $\sigma''(y) = \sigma(y)+1 > 0$.

$\sigma''(x)\geq 0 \quad \sigma' = \sigma''[0/x, \sigma''(x)+\sigma''(y)/y, \sigma(z)\cdot(\sigma(y)+\sigma(x))!/\sigma(y)!/z]$ dall'ipotesi indutt. essendo vere le premesse

$$\sigma'(x) \overset{?}{\geq} 0 \quad \sigma' \overset{?}{=} \sigma[0/x, \sigma(x)+\sigma(y)/y, \sigma(z)(\sigma(y)+\sigma(x))!/\sigma(y)!/z]$$

$$\sigma''(x) = \sigma(x)-1 \qquad \sigma(x)\geq 1 \geq 0 \quad \text{CVD.}$$

$$\sigma' = \sigma[\sigma(x)-1/x, \sigma(y)+1/y, \sigma(z)\sigma(y)/z][0/x, \sigma(x)-1+\sigma(y)+1/y,$$
$$\sigma(z)(\sigma(y)+1)(\sigma(y)+1+\sigma(x)-1)!/(\sigma(y)+1)!/z]$$

$$= \sigma[0/x, \sigma(x)+\sigma(y)/y, \sigma(z)(\sigma(y)+\sigma(x))!/\sigma(y)!/z] \quad \text{CVD.}$$

b) Usando la regola di inferenza:
$$\frac{\forall\sigma\in S.\ \langle c,\sigma\rangle\to\sigma' \Rightarrow \sigma'\in S \wedge \mathcal{B}[\![b]\!]\sigma' = true \qquad \sigma_1\in S}{\langle \text{while } b \text{ do } c, \sigma_1\rangle \not\to}$$

$$S = \{\sigma \mid \sigma(0) < 0\} \qquad \sigma\in S \quad \langle c,\sigma\rangle \to \sigma' \quad \sigma'\in S? \quad \sigma' = \sigma[x-1/x] \text{ CVD}$$

$$\sigma\in S' \quad \mathcal{B}[\![x\neq 0]\!]\sigma \overset{?}{=} true \qquad \text{CVD.}$$

Esercizio 2

Essendo $\sqsubseteq = <^*$ le proprietà riflessiva e simmetrica valgono per costruzione. La proprietà antisimmetrica vale in quanto $<$ è una relazione aciclica (addirittura ben fondata) e l'operazione di chiusura transitiva non introduce cicli.

Σ_n è l'alfabeto delle operazioni n-adiche

Caso $c \in \Sigma_0$ e $f \in \Sigma_n$

In tal caso T_Σ è un insieme infinito. La catena

$$t_0 = c \sqsubseteq t_1 = f(t_0 \dots t_0) \sqsubseteq t_2 = f(t_1 \dots t_1) \sqsubseteq \dots$$

non ha nessun maggiorante in T_Σ.

Caso $\Sigma_0 = \phi$

In tal caso T_Σ è vuoto. Quindi completo e senza bottom.

Caso $\Sigma_n = \phi$, $n \neq 0$

Allora $\sqsubseteq$ è un ordinamento discreto $(t_1 \sqsubseteq t_2 \Rightarrow t_1 = t_2)$ quindi con catene finite (di un solo elemento) quindi completo.

Caso $\Sigma_0 = \{c\}$ singleton

L'ordinamento $(T_\Sigma, \sqsubseteq)$ ha c come bottom

Caso Σ_0 non singleton

$(T_\Sigma, \sqsubseteq)$ non ha minimo: vuoto se $\Sigma_0 = \emptyset$ e più di un elemento minimale se Σ_0 ha due o più elementi.

Si ricordi che T_Σ è generato dalla regola di inferenza:

$$\frac{t_i \in T_\Sigma \quad i = 1, \dots, n \quad f \in \Sigma_n}{f(t_1, \dots, t_n) \in T_\Sigma}$$

Esercizio 3

a) $$\frac{t_1 : \sigma \to \tau_1 \qquad t_2 : \sigma \to \tau_2}{\langle t_1, t_2 \rangle : \sigma \to \tau_1 * \tau_2}$$

b) $$\langle t_1, t_2 \rangle \to \lambda x : \sigma . ((t_1\, x), (t_2\, x)) \qquad x \notin FV(t_1, t_2)$$

c) $$[\![t_1, t_2]\!]\rho = \lfloor \lambda d \in (V_\sigma)_\perp . \lfloor (\mathrm{let}\ \varphi \Leftarrow [\![t_1]\!]\rho . \varphi d, \ \mathrm{let}\ \varphi \Leftarrow [\![t_2]\!]\rho . \varphi d) \rfloor \rfloor$$

a)

$$\underbrace{(\underbrace{\langle \underbrace{t_1}_{\sigma \to \tau_1}, \underbrace{t_2}_{\sigma \to \tau_2} \rangle}_{\sigma \to \tau_1 * \tau_2} \ \underbrace{t}_{\sigma})}_{\tau_1 * \tau_2} \qquad \underbrace{(\underbrace{(\underbrace{t_1}_{\sigma \to \tau_1} \ \underbrace{t}_{\sigma})}_{\tau_1}, \underbrace{(\underbrace{t_2}_{\sigma \to \tau_2} \ \underbrace{t}_{\sigma})}_{\tau_2})}_{\tau_1 * \tau_2}$$

b) operazionale

$$(\langle t_1, t_2 \rangle\, t) \to c \Leftarrow \langle t_1, t_2 \rangle \to \lambda x . t' \quad t'[t/x] \to c \Leftarrow$$

$$((t_1\, x), (t_2\, x))[t/x] \to c \Leftarrow ((t_1\, t), (t_2\, t)) \to c \qquad \text{OK}$$

denotazionale

$$[\![(\langle t_1, t_2 \rangle\, t)]\!]\rho = \mathrm{let}\ \theta \Leftarrow \lfloor \lambda d . \lfloor (\mathrm{let}\ \varphi \Leftarrow [\![t_1]\!]\rho . \varphi d, \ \mathrm{let}\ \varphi \Leftarrow [\![t_2]\!]\rho . \varphi d) \rfloor \rfloor . \theta([\![t]\!]\rho)$$

$$= \lfloor (\mathrm{let}\ \varphi \Leftarrow [\![t_1]\!]\rho . \varphi([\![t]\!]\rho), \ \mathrm{let}\ \varphi \Leftarrow [\![t_2]\!]\rho . \varphi([\![t]\!]\rho)) \rfloor$$

$$= [\![((t_1\, t), (t_2\, t))]\!]$$

Exercise 4

$[\![\mu x.((p \wedge \Box x) \vee (\neg p \wedge \Diamond x))]\!]\rho =$

$= fix\ \lambda S.(\rho p \cap \{\sigma \mid \forall \sigma'.\ \sigma \to \sigma',\ \sigma' \in S)$

$\cup\ (\neg \rho p \cap \{\sigma \mid \exists \sigma'.\ \sigma \to \sigma',\ \sigma' \in S)$

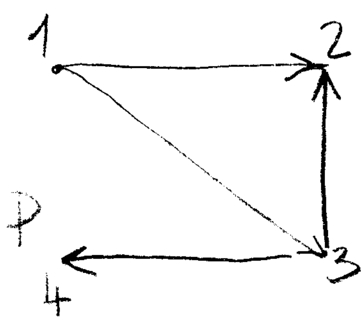
$S_0 = \phi$

$S_1 = \{4\} \cap \{2,4\} \cup \{1,2,3\} \cap \phi = \{4\}$

$S_2 = \{4\} \cap \{2,3,4\} \cup \{1,2,3\} \cap \{3\} = \{3,4\}$

$S_3 = \{4\} \cap \{2,3,4\} \cup \{1,2,3\} \cap \{1,3\} = \{1,3,4\}$

$S_4 = \{4\} \cap \{2,3,4\} \cup \{1,2,3\} \cap \{1,3\} = \{1,3,4\} = S_3 = fix$



Exercise 5

Agent P_1 can extrude new names $y_1, y_2, \dots$ and create corresponding agents

$Q_1 = !(\bar{x}y_1.nil)$, $Q_2 = !(\bar{x}y_2.nil), \dots$

which are able to communicate on channel x any number of times the new names $y_1, y_2, \dots$.

Agent P_2 is able to receive (any number of times) on x pairs of such new names, and to create messages $\bar{y}_i y_j.nil$, which can be sent on channels y_i.

$$(x)(P_1|P_2) \xrightarrow{\bar{z}(y_1)} (x)(P_1|P_2|Q_1) \xrightarrow{\bar{z}(y_2)} (x)(P_1|P_2|Q_1|Q_2)$$

$$\xrightarrow{\bar{z}(y_3)} (x)(P_1|P_2|Q_1|Q_2|Q_3) \xrightarrow{\tau} (x)(P_1|P_2|Q_1|Q_2|Q_3|x(y')\bar{y}_1 y'.nil)$$

$$\xrightarrow{\tau} (x)(P_1|P_2|Q_1|Q_2|Q_3|\bar{y}_1 y_1.nil)$$

$$\xrightarrow{\tau^{16}} (x)(P_1|P_2|Q_1|Q_2|Q_3|\bar{y}_1 y_1.nil|\bar{y}_1 y_2.nil|\bar{y}_1 y_3.nil|$$
$$\bar{y}_2 y_1.nil|\bar{y}_2 y_2.nil|\bar{y}_2 y_3.nil|$$
$$\bar{y}_3 y_1.nil|\bar{y}_3 y_2.nil|\bar{y}_3 y_3.nil = P'$$