Examen Programmation I (2017-18)

On insistera sur la correction et la clarté des réponses.

⋆ ⋆ ⋆

1 Théorie des domaines

Vous avez vu en TD qu'un dcpo X est *algébrique* si et seulement si pour tout $z \in X$, il existe une famille dirigée $(z_i)_{i\in I}$ d'éléments *finis* telle que $\sup_{i\in I} z_i = z$. Un élément z est *fini* si et seulement si pour toute famille dirigée $(y_j)_{j\in J}$ telle que $z \leq \sup_{j\in J} y_j$, alors il existe un $j \in J$ tel que $z \leq y_j$.

Un *inf-demi-treillis* est un ensemble ordonné dans lequel tout couple d'éléments x, y a une borne inférieure, que l'on notera $x \wedge y$.

On rappelle qu'un dcpo X est *conjonctif* si et seulement si c'est un inf-demi-treillis et pour tout $x \in X$, la fonction $y \mapsto x \wedge y$ est Scott-continue. Dans ce cas, la fonction $(x, y) \mapsto x \wedge y$ est Scott-continue.

1. (7 lignes) Montrer que tout inf-demi-treillis qui est aussi un dcpo algébrique est conjonctif. Indication : si z est fini et inférieur ou égal à $x \wedge \sup_{j\in J} y_j$, pourquoi z est-il inférieur ou égal à $\sup_{j\in J}(x \wedge y_j)$?

2 Real PCF$^-$

On redonne les sémantiques dénotationnelle et opérationnelle de Real PCF$^-$ dans l'annexe (Section 4). On rappelle que, pour tout $u : \tau$, $[\![u]\!]$ est une fonction bien définie, Scott-continue de $Env \stackrel{\text{def}}{=} \prod_{x_\sigma \text{ variable}} [\![\sigma]\!]$ vers $[\![\tau]\!]$.

2. (14 lignes) Montrer que la construction $u > 0$ de RealPCF$^-$ est redondante. De façon explicite, proposer une définition d'une expression Real PCF$^-$ **nonzero**, de type $\mathtt{I} \to \mathtt{unit}$, qui n'utilise pas d'expression de la forme $u > 0$, et dont la sémantique $[\![\mathbf{nonzero}]\!]\rho$ soit la fonction qui à 0 associe $\bot$ et à tout $a \in \mathcal{I}$ non nul associe $\top$. On demande la preuve de cette affirmation.
3. (3 lignes) Montrer que l'avant-avant-dernière règle de la sémantique opérationnelle est correcte, au sens où $[\![\mathtt{pif}\ u\ \mathtt{then}\ v\ \mathtt{else}\ *]\!]\rho = [\![v]\!]\rho$ pour tout $\rho \in Env$.
4. (5 lignes) On aimerait montrer l'adéquation. Bien entendu, c'est trop difficile et je ne le demande pas. Mais voici un argument qui pourrait faire partie de la preuve. On

considère un programme Real PCF$^-$ de la forme $\texttt{letrec}\ x_\sigma = u\ \texttt{in}\ v$, de type `unit`. Montrer que si $[\![\texttt{letrec}\ x_\sigma = u\ \texttt{in}\ v]\!]\rho \neq \bot$, alors il existe un entier $n \in \mathbb{N}$ tel que :

$$[\![\texttt{letrec}\ x_\sigma = u\ \texttt{in}\ v]\!]\rho = g(f^n(\bot)),$$

où l'on a utilisé les abréviations $g(V) = [\![v]\!](\rho[x_\sigma \mapsto V])$ et $f(V) = [\![u]\!](\rho[x_\sigma \mapsto V])$. (Le $\bot$ en argument de f^n est celui de $[\![\sigma]\!]$.) Ceci exprime qu'une définition récursive (de x_σ) utilisée dans un calcul terminant (v) de type `unit` ne sera « dépliée » que n fois.

5. (2 lignes, avec contre-exemple) Pourquoi l'argument de la question précédente ne fonctionne-t-il pas si $\texttt{letrec}\ x_\sigma = u\ \texttt{in}\ v$ est de type `I` ?
6. (5 lignes) On rappelle que $\dot{0} \stackrel{\text{def}}{=} \texttt{letrec}\ x_\texttt{I} = 0.x_\texttt{I}\ \texttt{in}\ x_\texttt{I}$. Montrer qu'il n'existe aucune dérivation dans la sémantique opérationnelle de :

$$\texttt{tl}_0(\texttt{pif}\ \dot{0} > 1/2\ \texttt{then}\ 1.\dot{0}\ \ \texttt{else}\ \ 0.1.1.\dot{0}) > 1/2 \to^* *.$$

   On pourra poser $Z \stackrel{\text{def}}{=} \texttt{letrec}\ x_\texttt{I} = 0.x_\texttt{I}\ \texttt{in}\ 0.x_\texttt{I}$.
7. (8 lignes) Que peut-on en conclure pour l'adéquation au type `unit` ? Justifier.
8. (9 lignes) Des suggestions pour compléter la sémantique opérationnelle ?

3 Typage

On suppose dorénavant qu'une même variable Real PCF$^-$ est toujours étiquetée avec le même type : si l'on rencontre x_σ et x_τ, alors $\sigma = \tau$. Ceci revient à dire que le nom x des variables suffit à les distinguer.

On considère le langage Real PCF^{--}, qui est juste Real PCF$^-$ mais sans aucun indice de type. Par exemple, $\texttt{fn}\ x.u$ et $\texttt{letrec}\ x = u\ \texttt{in}\ v$ sont les expressions Real PCF^{--} correspondant à $\texttt{fn}\ x_\sigma.u$ et à $\texttt{letrec}\ x_\sigma = u\ \texttt{in}\ v$, respectivement.

Formellement, notons E la fonction d'*effacement* des types, définie par $E(\texttt{letrec}\ x_\sigma = u\ \texttt{in}\ v) \stackrel{\text{def}}{=} \texttt{letrec}\ x = E(u)\ \texttt{in}\ E(v)$, $E(\texttt{fn}\ x_\sigma.u) \stackrel{\text{def}}{=} \texttt{fn}\ x.E(u)$, etc.

On dira qu'une expression Real PCF^{--} u, est *typable*, de type τ, si et seulement s'il existe une expression Real PCF$^-$ u', de type τ, telle que $E(u') = u$.

9. (1 ligne) Toutes les expressions Real PCF^{--} sont-elles typables ? Justifier.
10. (1 ligne) Le type d'une expression Real PCF^{--} typable est-il unique ? Justifier.

On souhaite construire un algorithme de typage de Real PCF^{--} à la Hindley. On supposera les expressions rectifiées (toute variable est liée à au plus un endroit). Etant donné une expression Real PCF^{--} u_0, on crée une variable de type α_t pour chaque sous-expression t de u_0, et un ensemble d'équations à résoudre.

Par exemple, pour $t = uv$, on créera l'équation $\alpha_t = \alpha_u \to \alpha_{uv}$.

11. (1 ligne) Quelles équations créera-t-on pour $t = \texttt{letrec}\ x = u\ \texttt{in}\ v$?
12. (1 ligne) Et pour $t = \texttt{pif}\ u\ \texttt{then}\ v\ \ \texttt{else}\ \ w$?
13. (1 ligne) Une fois toutes les équations produites pour notre expression u_0, comment pouvons-nous décider, en temps polynomial, si u_0 est typable ?

4 Annexe

Les types de Real PCF$^-$:

$$\begin{array}{rcl} \sigma, \tau, \ldots & ::= & \texttt{unit} \\ & | & \texttt{I} \\ & | & \sigma \to \tau \end{array}$$

$\mathbb{S} = \{\bot, \top\}$ avec $\bot < \top$. $\mathcal{I} = [0,1]$, avec l'ordre usuel.

$$[\![\texttt{unit}]\!] = \mathbb{S} \qquad [\![\texttt{I}]\!] = \mathcal{I} \qquad [\![\sigma \to \tau]\!] = [[\![\sigma]\!] \to [\![\tau]\!]].$$

$$\begin{array}{rl} [\![x_\tau]\!]\rho = & \rho(x_\tau) \\ [\![uv]\!]\rho = & [\![u]\!]\rho([\![v]\!]\rho) \\ [\![\texttt{fn}\ x_\sigma.u]\!]\rho = & (V \in [\![\sigma]\!] \mapsto [\![u]\!](\rho[x_\sigma \mapsto V])) \\ [\![\texttt{letrec}\ x_\sigma = u\ \texttt{in}\ v]\!] = & [\![v]\!](\rho[x_\sigma \mapsto [\![\texttt{rec}\ x_\sigma = u]\!]\rho]) \\ [\![\texttt{rec}\ x_\sigma = u]\!]\rho = & \mathrm{lfp}(V \in [\![\sigma]\!] \mapsto [\![u]\!](\rho[x_\sigma \mapsto V])) \end{array}$$

$$[\![0.u]\!]\rho = add_0([\![u]\!]\rho) \qquad [\![1.u]\!]\rho = add_1([\![u]\!]\rho)$$

$$[\![\texttt{tl}_0 u]\!]\rho = rem_0([\![u]\!]\rho) \qquad [\![\texttt{tl}_1 u]\!]\rho = rem_1([\![u]\!]\rho)$$

$$[\![\texttt{pif}\ u\ \texttt{then}\ v\ \texttt{else}\ w : \tau]\!]\rho = \begin{cases} [\![v]\!]\rho & \text{si } [\![u]\!]\rho = \top \\ [\![v]\!]\rho \wedge [\![w]\!]\rho & \text{si } [\![u]\!]\rho = \bot \end{cases}$$

$$[\![u > 1/2]\!]\rho = \begin{cases} \top & \text{si } [\![u]\!]\rho > 1/2 \\ \bot & \text{sinon} \end{cases} \qquad [\![u > 0]\!]\rho = \begin{cases} \top & \text{si } [\![u]\!]\rho > 0 \\ \bot & \text{sinon} \end{cases}$$

$$[\![*]\!]\rho = \top,$$

où $V \in X \mapsto f(V)$ désigne la fonction qui à tout V de X associe $f(V)$, et où :

$$add_0(a) = a/2 \qquad add_1(a) = (a+1)/2$$

$$rem_0(a) = \min(2a, 1) \qquad rem_1(a) = \max(2a-1, 0)$$

Contextes (contraintes de types omises) :

$$\begin{array}{rl} \mathcal{C} ::= & \_ \\ | & \mathcal{C} v \\ | & \texttt{tl}_0 \mathcal{C} \\ | & \texttt{tl}_1 \mathcal{C} \\ | & \mathcal{C} > 1/2 \\ | & \mathcal{C} > 0 \\ | & \texttt{pif}\ \mathcal{C}\ \texttt{then}\ v\ \texttt{else}\ w \\ | & \texttt{pif}\ u\ \texttt{then}\ \mathcal{C}\ \texttt{else}\ w \\ | & \texttt{pif}\ u\ \texttt{then}\ v\ \texttt{else}\ \mathcal{C} \end{array}$$

Sémantique opérationnelle. On n'applique une règle que sous un contexte $\mathcal{C}$ de la forme ci-dessus, i.e., $u \to v$ si et seulement si $u = \mathcal{C}[\ell]$ et $v = \mathcal{C}[r]$, où $\mathcal{C}$ est un contexte (les types étant

respectés), et $\ell \to r$ est l'une des règles ci-dessous.

$$
\begin{array}{rcll}
(\texttt{fn}\ x_\sigma.u)v & \to & u[x_\sigma := v] & \\
\texttt{letrec}\ x_\sigma = u\ \texttt{in}\ v & \to & v[x_\sigma := \texttt{letrec}\ x_\sigma = u\ \texttt{in}\ u] & \\
\texttt{tl}_a(a.u) & \to & u & (a \in \{0,1\}) \\
\texttt{tl}_0(1.u) & \to & \dot{1} & \\
\texttt{tl}_1(0.u) & \to & \dot{0} & \\
(1.u) > 1/2 & \to & u > 0 & \\
(1.u) > 0 & \to & * & \\
(0.u) > 0 & \to & u > 0 & \\
\texttt{pif}\ *\ \texttt{then}\ v\ \texttt{else}\ w & \to & v & \\
\texttt{pif}\ u\ \texttt{then}\ v\ \texttt{else}\ * & \to & v & \\
\texttt{pif}\ u\ \texttt{then}\ 0.v\ \texttt{else}\ 1.w & \to & 0.v & \\
\texttt{pif}\ u\ \texttt{then}\ a.v\ \texttt{else}\ a.w & \to & a.(\texttt{pif}\ u\ \texttt{then}\ v\ \texttt{else}\ w) & (a \in \{0,1\})
\end{array}
$$