Examen Programmation I (2022-23)

Tous documents écrits autorisés. Pas d'ordinateur, de tablette, de smartphone.

On insistera sur la correction et la clarté des réponses.

Toute affirmation devra être justifiée explicitement, par un théorème du cours, par un numéro de question, par une référence à une règle.

N'inventez pas vos propres notations, et réutilisez les miennes, même si elles ne vous plaisent pas.

Je veux une copie au propre, pas un brouillon : pas de rature, notamment. L'orthographe, et plus généralement la présentation, est importante.

Je corrigerai vraisemblablement toutes les copies en corrigeant d'abord toutes les questions 1 de tout le monde, puis toutes les questions 2, et ainsi de suite : ne comptez pas sur le fait que je me souvienne de ce que vous avez écrit dans une question précédente.

J'appellerai *fonction* ce qu'on appelle plus communément une application, c'est-à-dire une fonction totale. On ne les confondra pas avec les *noms de fonction* (voir plus bas).

⋆ ⋆ ⋆

Le sujet porte sur un langage permettant le calcul sur des *réels exacts*. Contrairement aux nombres en virgule flottante, les calculs sur ces réels ne font aucune erreur d'arrondi. Une représentation agréable de ces réels (entre -1 et 1) est sous forme de liste infinie (paresseuse) $x_1 : x_2 : \cdots : x_k : \cdots$ de nombres dans $\{-1, 0, 1\}$. Une telle liste représente $\sum_{k=1}^{+\infty} x_k/2^k$; un nombre est en général représenté par plusieurs listes infinies : la représentation est redondante.

On peut coder toutes les fonctions usuelles sur les réels sur cette représentation. Si ça vous intéresse, je vous recommande : Weng Kin Ho, *Exact Real Calculator for Everyone*, Proceedings of the 1st Asian Technology Conference in Mathematics (ATCM'13), Mumbai, India, 2013, Yang, W.-C. and Majweski, M. and Alwis, T. and Rana, I. K., editors.

On va étudier un sous-langage de Haskell qui sera suffisant pour écrire ce genre de programmes : Hask$\mathbb{R}$. Nous en donnerons deux variantes : une version sans aucune annotation de type (Section 1), et une version où toutes les variables, tous les noms de fonctions ont un type unique (Section 2).

1 Le langage Hask$\mathbb{R}$: typage

Les types de Hask$\mathbb{R}$ sont `I` et `int` (rien d'autre). Les termes sont décrits, avec les conventions usuelles, par :

$M, N, P, \cdots$		termes
	$::= x, y, z, \cdots$	variables
	$\mid f(M_1, M_2, \cdots, M_n)$	application de nom de fonction ($n \in \mathbb{N}$)
	$\mid \dot{n}$	constantes entières ($n \in \mathbb{Z}$)
	$\mid M \dot{+} N$	addition entière
	$\mid M : N$	constructeur de liste
	$\mid$ `hd` M	premier élément de liste
	$\mid$ `tl`M	reste de liste.

Le constructeur de liste $M : N$ serait noté en $M :: N$ en OCaml ; nous choisissons ici la syntaxe Haskell. Les symboles f des applications sont pris dans un ensemble Fun, supposé disjoint de l'ensemble Var des variables x, y, z, ..., et appelé l'ensemble des *noms de fonctions*.

Un *programme* π est intuitivement une liste de déclarations de fonctions (récursives, et même mutuellement récursives). Formellement, π est une fonction d'un sous-ensemble fini de Fun vers des *définitions* de fonctions : pour chaque nom de fonction f dans le domaine de π, $\pi(f)$ est un couple $\langle (x_1, x_2, \cdots, x_n), M \rangle$ où M est un terme et où x_1, x_2, ..., x_n sont des variables distinctes qui contiennent toutes les variables qui apparaissent dans M.

Les règles de typage sont :

$$\frac{}{\Gamma, x\colon \tau \vdash x\colon \tau}\ (Var :) \qquad \frac{\Gamma \vdash M_1\colon \tau_1 \cdots \Gamma \vdash M_n\colon \tau_n}{\Gamma \vdash f(M_1, \cdots, M_n)\colon \tau}\ (App :)$$
si Γ contient $f\colon \tau_1, \cdots, \tau_n \to \tau$

$$\frac{}{\Gamma \vdash \dot{n}\colon \texttt{int}}\ (Cst :) \qquad \frac{\Gamma \vdash M\colon \texttt{int} \quad \Gamma \vdash N\colon \texttt{int}}{\Gamma \vdash M \dot{+} N\colon \texttt{int}}\ (Plus :)$$

$$\frac{\Gamma \vdash M\colon \texttt{int} \quad \Gamma \vdash N\colon \texttt{I}}{\Gamma \vdash M : N\colon \texttt{I}}\ (Cons :) \quad \frac{\Gamma \vdash M\colon \texttt{I}}{\Gamma \vdash \texttt{hd}\ M\colon \texttt{int}}\ (Hd :) \quad \frac{\Gamma \vdash M\colon \texttt{I}}{\Gamma \vdash \texttt{tl} M\colon \texttt{I}}\ (Tl :)$$

Dans ces règles, τ, τ_1, ..., τ_n sont des types, donc soit `int` soit `I`. Un contexte Γ est un ensemble de *liaisons* de la forme $x\colon \tau$ ($x \in Var$) ou $f\colon \tau_1, \cdots, \tau_n \to \tau$ ($f \in Fun$), les mêmes variables ou noms de fonctions n'apparaissant qu'au plus une fois à gauche du symbole de typage :.

On dit qu'un programme π est *bien typé* si et seulement s'il existe un contexte Γ tel que, pour tout f dans le domaine de π, et si $\pi(f) = \langle (x_1, x_2, \cdots, x_n), M\rangle$, alors Γ contient une liaison de la forme $f\colon \tau_1, \cdots, \tau_n \to \tau$ (avec le même n) et l'on peut dériver :

$$\Gamma, x_1\colon \tau_1, \cdots, x_n\colon \tau_n \vdash M\colon \tau$$

dans le système de typage ci-dessus. Un tel contexte Γ sera appelé un *typage* de π.

Question 1 Donner un algorithme efficace (en temps polynomial) permettant de décider si π est bien typé, et le cas échéant, en retourne un typage principal. Je n'ai pas besoin de preuve de correction, mais l'algorithme devra être correct.

> *On utilise un algorithme à la Hindley-Milner. On crée une variable de typage α_M pour tout sous-terme apparaissant dans π, pour tout f dans le domaine de π, et pour toute variable x_i dans la liste des paramètres formels $(x_1, \cdots, x_n)$ de $\pi(f)$ pour chacun de ces symboles f.*
>
> *On produit ensuite les équations :*
>
> — *pour chaque symbole f dans le domaine de π, si $\pi(f) = \langle (x_1, x_2, \cdots, x_n), M\rangle$, l'équation $\alpha_f = (\alpha_{x_1}, \cdots, \alpha_{x_n} \to \alpha_M)$;*
>
> — *(Var :) rien pour les sous-termes qui sont des variables ;*
>
> — *(App :) pour chaque sous-terme $f(M_1, \cdots, M_n)$, l'équation $\alpha_f = (\alpha_{M_1}, \cdots, \alpha_{M_n} \to \alpha_{f(M_1, \cdots, M_n)})$;*

— (*Cst* :) *pour chaque sous-terme* $\dot{n}$*, l'équation* $\alpha_{\dot{n}} = \texttt{int}$ *;*

— (*Plus* :) *pour chaque sous-terme* $M \dot{+} N$*, les équations* $\alpha_M = \texttt{int}$*,* $\alpha_N = \texttt{int}$*, et* $\alpha_{M \dot{+} N} = \texttt{int}$ *;*

— (*Cons* :) *pour chaque sous-terme* $M : N$*, les équations* $\alpha_M = \texttt{int}$*,* $\alpha_N = \texttt{I}$*, et* $\alpha_{M:N} = \texttt{I}$ *;*

— (*Hd* :) *pour chaque sous-terme* $\texttt{hd}\, M$*, les équations* $\alpha_M = \texttt{I}$ *et* $\alpha_{\texttt{hd}\, M} = \texttt{int}$ *;*

— (*Tl* :) *pour chaque sous-terme* $\texttt{tl}M$*, les équations* $\alpha_M = \texttt{I}$ *et* $\alpha_{\texttt{tl}M} = \texttt{I}$*.*

On résout ensuite ce système par un algorithme d'unification en temps polynomial, par exemple celui du cours qui opère sur des formes triangulaires.

Un algorithme qui énumérerait les choix `I` *vs.* `int` *pour chaque variable est à exclure, car il serait en temps exponentiel.*

2 Le langage Hask$\mathbb{R}$: sémantique

On suppose dorénavant tous les programmes et termes que nous manipulerons bien typés, tous dans le même contexte de typage Γ. Chaque variable x a donc un type unique σ, ce que l'on explicitera parfois en écrivant x_σ à la place de x ; chaque nom de fonction f a un type unique $\tau_1, \cdots, \tau_n \to \tau$, ce que l'on explicitera parfois par la notation $f_{\tau_1,\cdots,\tau_n \to \tau}$; et tout terme a lui aussi un type uniquement déterminé.

On définit une sémantique opérationnelle à petits pas de Hask$\mathbb{R}$ par les règles de la figure 1. Les configurations sont juste les termes du langage. $M[x_1 := N_1, x_2 := N_2, \cdots, x_n := N_n]$ est le terme obtenu à partir de M en remplaçant simultanément chaque occurrence d'une variable x_i par l'expression N_i correspondante.

Question 2 On dit qu'un terme M est *admissible* (pour le programme π) si et seulement si (il est bien typé et) s'il ne contient pas de variable, et si tous les noms de fonctions qui apparaissent dans M sont dans le domaine de π.

Une *p-valeur* dans notre cadre est un terme admissible M qui a terminé, c'est-à-dire telle que $M \to_\pi M'$ n'est dérivable pour aucun terme M'. Par exemple, n'importe quel terme admissible de la forme $M : N$ est une p-valeur.

$$\frac{}{f(N_1, N_2, \cdots, N_n) \to_\pi P[x_1 := N_1, x_2 := N_2, \cdots, x_n := N_n]}\ (App)$$
$$\text{où } \pi(f) = \langle (x_1, x_2, \cdots, x_n), P \rangle$$

$$\frac{M \to_\pi M'}{M \dot{+} N \to_\pi M' \dot{+} N}\ (\dot{+}L) \quad \frac{N \to_\pi N'}{M \dot{+} N \to_\pi M \dot{+} N'}\ (\dot{+}R) \quad \frac{}{\dot{m} \dot{+} \dot{n} \to_\pi \dot{p}}\ (\dot{+})$$
$$\text{où } p = m + n$$

$$\frac{M \to_\pi M'}{\texttt{hd}\ M \to_\pi \texttt{hd}\ M'}\ (\texttt{hd}\ L) \quad \frac{}{\texttt{hd}\ (M : N) \to_\pi M}\ (\texttt{hd})$$

$$\frac{M \to_\pi M'}{\texttt{tl}\ M \to_\pi \texttt{tl}\ M'}\ (\texttt{tl}\ L) \quad \frac{}{\texttt{tl}\ (M : N) \to_\pi N}\ (\texttt{tl})$$

FIGURE 1 – Sémantique opérationnelle de Hask$\mathbb{R}$

Répondre aux deux sous-questions suivantes. Je n'ai pas besoin de justification, mais le résultat devra être correct.

(a) Quelles sont les p-valeurs de type `I` ?

Juste celles de la forme $M : N$, avec M et N des termes admissibles quelconques, de types `int` *et* `I` *respectivement.*

Par typage, les seules autres termes de type `I` *sont les variables $x_{\texttt{I}}$, les applications $f(M_1, \cdots, M_n)$ qui sont de type* `I`*, les termes $M : N$, et les termes $\texttt{tl}M$. Il n'y a pas de variable dans un terme admissible. Par admissibilité aussi, $f(M_1, \cdots, M_n)$ n'a pas terminé (la règle (App) s'applique). Il ne reste donc que les termes de la forme $M : N$ ou $\texttt{tl}M$. Mais s'il existait une p-valeur de la forme $\texttt{tl}M$, il en existerait une de taille minimale, et donc M serait aussi un terme de type* `I`*, mais qui ne pourrait pas être de la forme $\texttt{tl}M'$; on aurait donc $M = M' : N'$; alors on aurait $\texttt{tl}(M' : N') \to_\pi N'$ par (*`tl`*), ce qui contredirait le fait que $\texttt{tl}M$ est une p-valeur. Donc les seules p-valeurs de type* `I` *sont de la forme $M : N$.*

Réciproquement, tout terme de la forme $M : N$ de type `I` *avec M et N admissibles est admissible, et aucune règle ne s'y applique. C'est donc bien une p-valeur de type* `I`*.*

(b) Quelles sont les p-valeurs de type `int` ?

$$
\begin{array}{rcl}
[\![x]\!]\rho & = & \rho(x) \\
[\![f(N_1, N_2, \cdots, N_n)]\!]\rho & = & \rho(f)([\![N_1]\!]\rho, [\![N_2]\!]\rho, \cdots, [\![N_n]\!]\rho) \\
[\![\dot{n}]\!]\rho & = & n \\
[\![M \dot{+} N]\!]\rho & = & \begin{cases} [\![M]\!]\rho + [\![N]\!]\rho & \text{si } [\![M]\!]\rho \neq \bot \text{ et } [\![N]\!]\rho \neq \bot \\ \bot & \text{sinon} \end{cases} \\
[\![M : N]\!]\rho & = & push([\![M]\!]\rho, [\![N]\!]\rho) \\
[\![\texttt{hd}\ M]\!]\rho & = & hd\ [\![M]\!]\rho \\
[\![\texttt{tl}\ M]\!]\rho & = & tl\ [\![M]\!]\rho
\end{array}
$$

FIGURE 2 – Sémantique dénotationnelle de Hask$\mathbb{R}$

Juste celles de la forme $\dot{n}$.

On raisonne similairement. Par admissibilité, ces p-valeurs ne peuvent être ni des variables ni des applications. Par typage, elles peuvent être de la forme $\dot{n}$, ou $M \dot{+} N$, ou `hd` *M. Mais un terme de taille minimale de l'une des deux dernières formes serait soit de la forme $\dot{m} \dot{+} \dot{n}$ (impossible : la règle $(\dot{+})$ s'appliquerait), soit de la forme* `hd` *$(M' : N')$, par la question 3.(a) (impossible : la règle* (`tl`) *s'appliquerait).*

Réciproquement, aucune règle ne s'applique à un terme de la forme $\dot{n}$.

On définit une sémantique dénotationnelle comme suit. On pose, pour les types :

$$[\![\texttt{int}]\!] = \mathbb{Z}_\bot \qquad [\![\texttt{I}]\!] = [\mathbb{N} \to \mathbb{Z}_\bot].$$

On rappelle que $\mathbb{Z}_\bot$ est le dcpo plat formé de tous les entiers relatifs, incomparables deux à deux, plus un élément $\bot$ inférieur à tous les entiers. $\mathbb{N}$ est équipé de l'ordre $=$, donc $[\mathbb{N} \to \mathbb{Z}_\bot]$ n'est rien d'autre que le dcpo des suites infinies d'éléments de $\mathbb{Z}_\bot$, ordonnées composante par composante.

La sémantique elle-même est donnée en figure 2. Dans cette sémantique, on utilise des environnements ρ. Ce sont des fonctions qui à chaque variable x_σ associe une valeur de $[\![\sigma]\!]$ et à chaque nom de fonction $f_{\sigma_1,\sigma_2,\cdots,\sigma_n \to \tau}$ associe une fonction Scott-continue de $[\![\sigma_1]\!] \times [\![\sigma_2]\!] \times \cdots \times [\![\sigma_n]\!]$ vers $[\![\tau]\!]$.

Pour une liste infinie $\ell \in [\![\texttt{I}]\!]$, on notera $hd\ \ell$ l'élément $\ell(0)$ de $\mathbb{Z}_\bot$, et $tl\ \ell$ la liste définie par $tl\ \ell(n) = \ell(n+1)$ $(n \in \mathbb{N})$.

On notera aussi $push(m, \ell)$ la liste dont l'élément à l'indice 0 vaut m, et dont l'élément à l'indice $n+1$ est $\ell(n)$, pour tout $n \in \mathbb{N}$.

Question 3 Soit M un terme fixé. On souhaite montrer que la fonction qui à $\rho \in Env$ associe $[\![M]\!]\rho$ est Scott-continue. On notera cette fonction $[\![M]\!]$, plus brièvement.

(a) Sur quoi se fait la récurrence ?

Sur la structure, ou la taille, de M.

On fait ensuite une analyse de cas. Par exemple, si M est de la forme $N : P$, l'argument repose sur le fait que *push* est Scott-continue de $\mathbb{Z}_\perp \times [\mathbb{N} \to \mathbb{Z}_\perp]$ vers $[\mathbb{N} \to \mathbb{Z}_\perp]$.

(b) Montrer que *push* est Scott-continue.

La réponse la plus rapide est que push est en fait un isomorphisme d'ordre ($push(m,\ell) \leq push(m',\ell')$ si et seulement si $(m,\ell) \leq (m',\ell')$), et que tout isomorphisme d'ordre est Scott-continue. Mais ce n'a pas été vu en cours.

Monotonie. Si $m \leq m'$ et $\ell \leq \ell'$, alors on vérifie que $push(m,\ell) \leq push(m',\ell')$ en vérifiant que $push(m,\ell)(n) \leq push(m',\ell')(n)$ pour tout $n \in \mathbb{N}$:

- *si $n = 0$, ceci revient à $m \leq m'$;*
- *si $n \geq 1$, ceci revient à $\ell(n-1) \leq \ell'(n-1)$, ce qui est conséquence de $\ell \leq \ell'$.*

Ensuite, soit $(m_i, \ell_i)_{i\in I}$ une famille dirigée de supremum (m,ℓ). En particulier, $\sup_{i\in I} m_i = m$ et $\sup_{i\in I} \ell_i = \ell$, par le lemme 6 de `prog1_sem2.pdf`. On montre que $\sup_{i\in I} push(m_i,\ell_i)(n) = push(m,\ell)(n)$ comme suit. (On profite du fait que les sups dirigés sont calculés point à point, par le lemme 5 de `prog1_sem2.pdf`.)

- *si $n = 0$, ceci revient à $\sup_{i\in I} m_i = m$, ce que l'on a vu plus haut ;*
- *si $n \geq 1$, ceci revient à $\sup_{i\in I} \ell_i(n-1) = \ell(n-1)$, ce qui est une conséquence de $\sup_{i\in I} \ell_i = \ell$ et du fait que les sups dirigés sont calculés point à point.*

Dans la suite, nous admettrons tous les autres cas : la fonction $[\![M]\!]$ est Scott-continue.

Pour chaque nom de fonction $f_{\sigma_1,\sigma_2,\cdots,\sigma_n\to\tau}$, on notera D_f le dcpo $[[\![\sigma_1]\!] \times [\![\sigma_2]\!] \times \cdots \times [\![\sigma_n]\!] \to [\![\tau]\!]]$.

Pour un environnement ρ, et une définition de fonction $\pi(f) = \langle (x_{1\,\sigma_1}, x_{2\,\sigma_2}, \cdots, x_{n\,\sigma_n}), P\rangle$, on notera $Val(f,\rho)$ la fonction qui à tout $(V_1, V_2, \cdots, V_n) \in [\![\sigma_1]\!] \times [\![\sigma_2]\!] \times \cdots \times [\![\sigma_n]\!]$ associe $[\![P]\!][x_1 \mapsto V_1, x_2 \mapsto V_2, \cdots, x_n \mapsto V_n]$.

On note $Env = \prod_{f \text{ nom de fonction}} D_f \times \prod_{x_\sigma \text{ variable}} [\![\sigma]\!]$. C'est le dcpo des environnements. (C'est un dcpo car tout produit de dcpos est un dcpo. La démonstration se fait comme pour le cas des produits binaires, à savoir le lemme 6 de `prog1_sem2.pdf`, et on n'aura pas à le démontrer.) Étant donné un programme π, de domaine $\{f_1, f_2, \cdots, f_p\}$, on pose, pour tout $\rho \in Env$:

$$F_\pi(\rho) \stackrel{\text{def}}{=} \rho[f_1 \mapsto Val(f_1,\rho), f_2 \mapsto Val(f_2,\rho), \cdots, f_p \mapsto Val(f_p,\rho)].$$

On admettra que F_π est une fonction Scott-continue de Env vers Env. C'est une conséquence de la **Question 3**.

Question 4 Pourquoi F_π a-t-elle un plus petit point fixe ? On énoncera le théorème, et on expliquera pourquoi ses hypothèses sont vérifiées.

> *Par le théorème de Kleene : toute fonction Scott-continue f d'un dcpo pointé X vers lui-même a un plus petit point fixe, et il vaut $\sup_{n\in\mathbb{N}} f^n(\bot)$, où $\bot$ est le plus petit élément de X.*
>
> *Ceci s'applique car Env est un dcpo (ce qu'on vient d'admettre), et qu'il est pointé. En effet, $[\![\texttt{int}]\!]$ est pointé (il a un plus petit élément $\bot$), $[\![\texttt{I}]\!]$ est pointé (son plus petit élément est la fonction constante de valeur $\bot$), et tout produit de dcpos pointés est pointé (son plus petit élément est le uplet dont toutes les composantes sont les plus petits éléments de leurs domaines respectifs).*
>
> *Le théorème de Tarksi ne s'applique pas : D_π n'est pas un treillis complet ; par exemple, il n'existe pas de plus grand élément (c'est déjà le cas pour $[\![\texttt{int}]\!]$).*

Par la suite, on notera $\rho_\pi = \text{lfp}(F_\pi)$ ce plus petit point fixe.

Question 5 On admettra le résultat suivant : pour tout environnement ρ, si $\rho(x_1) = [\![N_1]\!]\rho$, $\rho(x_2) = [\![N_2]\!]\rho$, …, $\rho(x_n) = [\![N_n]\!]\rho$, alors :

$$\begin{aligned}&[\![M[x_1 := N_1, x_2 := N_2, \cdots, x_n := N_n]]\!]\rho\\ &= [\![M]\!](\rho[x_1 \mapsto [\![N_1]\!]\rho, x_2 \mapsto [\![N_2]\!]\rho, \cdots, [\![N_n]\!]\rho]).\end{aligned}$$

On souhaite montrer le résultat de correction suivant : si $M \to_\pi M'$ alors $[\![M]\!]\rho_\pi = [\![M']\!]\rho_\pi$. Montrez-le dans le cas où $M \to_\pi M'$ est obtenu par la règle (*App*).

On admettra dans la suite que le résultat est vrai dans le cas des autres règles aussi.

Par hypothèse, on a $M = f(N_1, N_2, \cdots, N_n)$, $\pi(f) = \langle (x_1, x_2, \cdots, x_n), P \rangle$, *et* $M' = M[x_1 := N_1, x_2 := N_2, \cdots, x_n := N_n]$.

De plus, $[\![M]\!]\rho_\pi = \rho_\pi(f)([\![N_1]\!]\rho_\pi, [\![N_2]\!]\rho_\pi, \cdots, [\![N_n]\!]\rho_\pi)$. *On utilise que* $\rho_\pi = \mathrm{lfp}(F_\pi)$ *est un point fixe de* F_π, *donc* $\rho_\pi(f) = F_\pi(\rho_\pi)(f)$, *et ceci est par définition est égal à* $Val(f, \rho_\pi)$.

Donc $[\![M]\!]\rho_\pi = Val(f, \rho_\pi)([\![N_1]\!]\rho_\pi, [\![N_2]\!]\rho_\pi, \cdots, [\![N_n]\!]\rho_\pi)$, *et par définition de* Val, *ceci vaut* $[\![P]\!][x_1 \mapsto [\![N_1]\!]\rho_\pi, x_2 \mapsto [\![N_2]\!]\rho_\pi, \cdots, x_n \mapsto [\![N_n]\!]\rho_\pi]$.

Comme les seules variables libres de P *sont parmi* x_1, ..., x_n, $[\![P]\!][x_1 \mapsto [\![N_1]\!]\rho_\pi, x_2 \mapsto [\![N_2]\!]\rho_\pi, \cdots, x_n \mapsto [\![N_n]\!]\rho_\pi] = [\![P]\!](\rho_\pi[x_1 \mapsto [\![N_1]\!]\rho_\pi, x_2 \mapsto [\![N_2]\!]\rho_\pi, \cdots, x_n \mapsto [\![N_n]\!]\rho_\pi])$. *(C'est une récurrence facile sur* P.*)*

Par le résultat admis, $[\![P]\!](\rho_\pi[x_1 \mapsto [\![N_1]\!]\rho_\pi, x_2 \mapsto [\![N_2]\!]\rho_\pi, \cdots, x_n \mapsto [\![N_n]\!]\rho_\pi])$ *vaut* $[\![M[x_1 := N_1, x_2 := N_2, \cdots, x_n := N_n]]\!]\rho_\pi$, *c'est-à-dire* $[\![M']\!]\rho_\pi$.

Question 6 Pour un terme admissible M, on obtient le terme admissible $\mathsf{Unfold}_\pi(M)$ en remplaçant chaque sous-terme de la forme $f(N_1, N_2, \cdots, N_n)$ dans M par $P[x_1 := N_1, x_2 := N_2, \cdots, x_n := N_n]$, où $\langle (x_1, x_2, \cdots, x_n), P \rangle$ est la définition $\pi(f)$ de f.

On peut montrer (et on l'admettra) que pour tout environnement ρ, $[\![\mathsf{Unfold}_\pi(M)]\!]\rho = [\![M]\!](F_\pi(\rho))$.

On pose $\rho_\perp$ l'environnement qui à toute variable et à tout nom de fonction associe le plus petit élément du dcpo associé.

Montrer que pour tout terme admissible M de type `int` tel que $[\![M]\!]\rho_\pi$ est un entier $m \in \mathbb{Z}$ (donc différent de $\perp$), il existe un entier $n \in \mathbb{N}$ tel que $[\![\mathsf{Unfold}_\pi^n(M)]\!]\rho_\perp = m$.

À titre d'indication, c'est un argument similaire à l'argument qui est au cœur de la preuve d'adéquation du langage Imp du cours.

L'environnement ρ_π *est le sup de la famille dirigée des* $F_\pi^n(\perp)$, *où* $\perp$ *est le plus petit environnement, par le théorème de Kleene. Comme* $[\![M]\!]$ *est continue par la* ***Question 5***, *on a* $\sup_{n \in \mathbb{N}} [\![\mathsf{Unfold}_\pi^n(M)]\!]\rho_\perp = m$. *Ceci est un sup dirigé dans un dcpo* plat, *il est donc atteint.*

Question 7 La conclusion de la question **Question 6** est-elle encore vraie lorsque M est de type `I` ? Autrement dit, supposons que $[\![M]\!]\rho_\pi$ soit une liste infinie $\ell = n_1 : n_2 : \cdots : n_k : \cdots$ d'entiers, tous différents de $\perp$:

existe-t-il un entier $n \in \mathbb{N}$ tel que $[\![\mathsf{Unfold}_\pi^n(M)]\!]\rho_\perp = \ell$?
Fournir une justification rapide si c'est le cas, un contre-exemple sinon.

C'est faux. L'argument de la question précédente ne s'applique par car $[\![\mathtt{I}]\!]$ n'est pas de hauteur finie.

A titre de contre-exemple, on suppose un programme π associant à un symbole f le couple $\langle(), 0 : f()\rangle$. Les valeurs successives de $[\![\mathsf{Unfold}_\pi^n(f())]\!]$ sont $\perp$, $0 : \perp$, $0 : 0 : \perp$, etc. Aucun ne vaut le sup, qui est la liste infinie contenant uniquement des zéros.

Question 8 On définit $\to_\perp$ comme étant égale à la relation $\to_\pi$ dans le cas où le programme π est vide. De façon équivalente, $M \to_\perp M'$ si et seulement si M se réécrit en M' en une étape par l'une des règles de la figure 1 autre que (*App*).

On souhaite montrer que pour tout terme N admissible de type `int` telle que $[\![N]\!]\rho_\perp \neq \perp$, il existe un $n \in \mathbb{Z}$ tel que $N \to_\perp^* \dot{n}$. Ceci n'étant pas démontrable directement, nous allons démontrer simultanément que pour tout terme admissible N :

(i) si N est de type `int` et $[\![N]\!]\rho_\perp \neq \perp$, alors il existe un entier $n \in \mathbb{Z}$ tel que $N \to_\perp^* \dot{n}$;

(ii) si N est de type `I`, alors pour tout entier $m \in \mathbb{N}$ tel que $[\![\mathtt{hd}\ \mathtt{tl}^m\ N]\!]\rho_\perp \neq \perp$, il existe un entier $n \in \mathbb{Z}$ tel que $\mathtt{hd}\ \mathtt{tl}^m\ N \to_\perp^* \dot{n}$.

Les sous-questions auxquelles vous avez à répondre sont les suivantes.

(a) Sur quoi se fait la récurrence ?

Sur la structure, ou la taille, de N.

(b) Traitez du cas où N est de la forme $P : Q$. Les autres cas seront admis.

Si $N = P : Q$, il est de type `I`, et on démontre (ii).

— *Si $m = 0$, on doit démontrer que si $[\![\mathtt{hd}\ (P : Q)]\!]\rho_\perp \neq \perp$, il existe un entier $n \in \mathbb{Z}$ tel que $\mathtt{hd}\ (P : Q) \to_\perp^* \dot{n}$. On a $\mathtt{hd}\ (P : Q) \to_\perp P$ par (`hd`). Par l'hypothèse de récurrence appliquée à P, partie (i) (puisque P est de type `int`, et puisque $[\![P]\!]\rho_\perp = [\![\mathtt{hd}\ (P : Q)]\!]\rho_\perp \neq \perp$), on a $P \to_\perp^* \dot{n}$ pour un certain $n \in \mathbb{Z}$.*

— *Si $m \geq 1$, on doit démontrer que si $[\![\mathtt{hd}\ \mathtt{tl}^m(P : Q)]\!]\rho_\perp \neq \perp$, il existe un $n \in \mathbb{Z}$ tel que $\mathtt{hd}\ \mathtt{tl}^m(P : Q) \to_\perp^* \dot{n}$. Or $[\![\mathtt{hd}\ \mathtt{tl}^m(P : Q)]\!]\rho_\perp = [\![\mathtt{hd}\ \mathtt{tl}^{m-1}Q]\!]\rho_\perp$. On peut donc*

appliquer l'hypothèse de récurrence sur Q, partie (ii) (puisque Q est de type `I`*), et l'on obtient* $\texttt{hd}\ \texttt{tl}^{m-1}Q \to_\bot^* \dot{n}$ *pour un certain entier $n \in \mathbb{Z}$. Puis on utilise la règle* (`tl`)*, $m-1$ fois la règle* (`tl` L)*, et une fois la règle* (`hd` L)*, pour affirmer que* $\texttt{hd}\ \texttt{tl}^m(P:Q) \to_\bot \texttt{hd}\ \texttt{tl}^{m-1}Q$.

Nous allons nous arrêter là. Nous ne sommes plus très loin de démontrer l'adéquation (si M est un terme admissible de type `int`, et que $[\![M]\!]\rho_\pi = m$ avec $m \neq \bot$, alors $M \to_\pi^* \dot{m}$). Il ne resterait plus qu'à démontrer que pour tout π-dépliage M^* de M, si $M^* \to_\bot^* \dot{m}$, alors $M \to_\pi^* \dot{m}$. Un *π-dépliage* de M est obtenu à partir de M en itérant la procédure consistant à remplacer un sous-terme de la forme $f(N_1, N_2, \cdots, N_n)$ dans M par $P[x_1 := N_1, x_2 := N_2, \cdots, x_n := N_n]$, où $\langle (x_1, x_2, \cdots, x_n), P\rangle$ est la définition $\pi(f)$ de f. Par la question **Question 6**, si $[\![M]\!]\rho_\pi = m$ avec $m \neq \bot$, $[\![\mathsf{Unfold}_\pi^n(M)]\!]\rho_\bot = m$ pour un certain n, donc $\mathsf{Unfold}_\pi^n(M) \to_\bot^* \dot{m}$ par la question **Question 8**. Or $\mathsf{Unfold}_\pi^n(M)$ est un dépliage de M, ce qui permet de conclure.