Examen Programmation I (2021-22)

Tous documents autorisés.

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.

Pour tout raisonnement par récurrence, on devra dire sur quoi est effectuée la récurrence.

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.

Je corrigerai très certainement 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.

Je me réserve le droit de ne pas chercher à comprendre une réponse illisible, insuffisamment claire, ou plus longue ou compliquée que nécessaire.

J'appellerai *fonction* ce qu'on appelle plus communément une application, c'est-à-dire une fonction totale.

1 Fermés de Scott

Un *fermé de Scott* d'un dcpo X est un sous-ensemble F de X tel que :

1. F est *clos par le bas* : pour tout $y \in F$, pour tout $x \leq y$, x est dans F ;
2. F est *stable par sups dirigés* : pour toute famille dirigée $(x_i)_{i \in I}$ d'éléments de F, $\sup_{i \in I} x_i$ est dans F.

On note $\mathcal{H}X$ l'ensemble des fermés de Scott de X, et on le munit de l'ordre d'inclusion $\subseteq$.

Il est facile de voir que toute intersection (finie ou infinie) de fermés de Scott est un fermé de Scott. Pour toute partie A de X, l'intersection des

fermés de Scott F qui contiennent A est donc aussi le plus petit fermé de Scott contenant A, et est appelé l'*adhérence* de A. On la notera $\overline{A}$. On a :

Lemme 1 *Si $A \subseteq B$ alors $\overline{A} \subseteq \overline{B}$.*

Démonstration. $\overline{B}$ est un fermé de Scott contenant A. Il contient donc le plus petit, $\overline{A}$. □

Question 1 Montrer que $\mathcal{H}X$ est un dcpo. Montrer aussi que le sup d'une famille dirigée $(F_i)_{i\in I}$ d'éléments de $\mathcal{H}X$ est l'adhérence $\overline{\bigcup_{i\in I} F_i}$ de son union.

Question 2 Montrer que, pour tous dcpos X et Y, pour toute fonction Scott-continue $f : X \to Y$, pour tout fermé de Scott F de Y, $f^{-1}(F)$ est un fermé de Scott de X.

De la **Question 2**, on peut déduire :

Lemme 2 *Pour toute fonction Scott-continue $f : X \to Y$ entre dcpos, pour tout sous-ensemble A de X, $f[\overline{A}] \subseteq \overline{f[A]}$.*

Démonstration. D'abord, $f^{-1}(\overline{f[A]})$ est un fermé de Scott par la **Question 2**, et il contient $f^{-1}(f[A])$, qui contient A. Donc il contient aussi le plus petit fermé de Scott contenant A, à savoir $\overline{A}$. Nous venons de démontrer $\overline{A} \subseteq f^{-1}(\overline{f[A]})$, et ceci est équivalent à $f[\overline{A}] \subseteq \overline{f[A]}$. (Si vous pensez avoir une démonstration plus simple, elle est peut-être fausse : voir l'annexe A.) □

On observe aussi :

Lemme 3 *Pour tout $F \in \mathcal{H}X$, l'ensemble $\downarrow F$ des fermés de Scott de X inclus dans F est lui-même un fermé de Scott de $\mathcal{H}X$ (un élément de $\mathcal{H}\mathcal{H}X$).*

Démonstration. Il est clos par le bas : si $F_1 \subseteq F_2$ et $F_2 \in \downarrow F$, alors F_1 est inclus dans F donc un élément de $\downarrow F$. Pour toute famille dirigée $(F_i)_{i\in I}$ d'éléments de $\downarrow F$ dans $\mathcal{H}X$, F_i est inclus dans F pour tout $i \in I$, donc F est un majorant de $(F_i)_{i\in I}$. Donc $\sup_{i\in I} F_i \subseteq F$, ce qui signifie que $\sup_{i\in I} F_i$ est dans $\downarrow F$. (On n'a pas besoin de savoir ce qu'est ce sup.) □

Question 3 Pour tous dcpos X et Y, et pour toute fonction Scott-continue $f : X \to \mathcal{H}Y$, on note $f^\dagger$ la fonction de $\mathcal{H}X$ vers $\mathcal{H}Y$ définie par $f^\dagger(F) \stackrel{\text{def}}{=} \overline{\bigcup_{x\in F} f(x)}$. Montrer que $f^\dagger$ est Scott-continue. C'est l'une des deux questions difficiles de l'examen (et donc qui compte le plus).

On dispose aussi d'une fonction $\eta : X \to \mathcal{H}X$ qui à tout $x \in X$ associe $\eta(x) \stackrel{\text{def}}{=} \downarrow x$, l'ensemble des points inférieurs ou égaux à x dans X. Il est

facile de voir que $\downarrow x$ est bien un fermé de Scott ; la démonstration est la même que celle du lemma 3. De plus, on admettra que η est Scott-continue. Dans la suite, on pourra utiliser sans les redémontrer les égalités suivantes :

$$\eta^\dagger(F) = F \qquad (F \in \mathcal{H}X) \qquad (1)$$

$$f^\dagger(\eta(x)) = f(x) \qquad (x \in X, f \in [X \to \mathcal{H}Y]) \qquad (2)$$

$$f^\dagger(g^\dagger(x)) = (f^\dagger \circ g)^\dagger(x) \qquad (x \in X, f \in [Y \to \mathcal{H}Z], g \in [X \to \mathcal{H}Y]) \qquad (3)$$

2 Le langage CImp

Considérons le langage CIMP dont la syntaxe est la suivante :

Commandes				Expressions		
c	::=	$x := e$	affectation	e ::=	x	variables
	$\mid$	`skip`	ne rien faire	$\mid$	$\dot{n}$	constante entière ($n \in \mathbb{Z}$)
	$\mid$	$c_1; c_2$	séquence	$\mid$	$e \dot{+} e$	addition
	$\mid$	`if` e `then` c_1 `else` c_2	conditionnelle	$\mid$	$\dot{-} e$	opposé
	$\mid$	`while` e `do` c	boucle while			
	$\mid$	`input` x	choix non déterministe			

La seule différence avec IMP est la nouvelle instruction `input` x, qui produit un entier de façon non déterministe, et affecte le résultat à x. On peut imaginer que `input` x demande à un utilisateur de taper un entier, et nous n'avons aucun moyen de prédire lequel.

La sémantique opérationnelle est la même que celle de IMP pour toutes les constructions autres que le choix non déterministe `input` x. Nous avons une seule nouvelle règle, la règle (11) ci-dessous.

$$(x := e \cdot C, \rho) \to (C, \rho[x \mapsto [\![e]\!]\rho]) \qquad (4)$$

$$(\texttt{skip} \cdot C, \rho) \to (C, \rho) \qquad (5)$$

$$(c_1; c_2 \cdot C, \rho) \to (c_1 \cdot c_2 \cdot C, \rho) \qquad (6)$$

$$(\texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \cdot C, \rho) \to (c_1 \cdot C, \rho) \quad \text{si } [\![e]\!]\rho \neq 0 \qquad (7)$$

$$(\texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \cdot C, \rho) \to (c_2 \cdot C, \rho) \quad \text{si } [\![e]\!]\rho = 0 \qquad (8)$$

$$(\texttt{while}\ e\ \texttt{do}\ c \cdot C, \rho) \to (c \cdot \texttt{while}\ e\ \texttt{do}\ c \cdot C, \rho) \text{ si } [\![e]\!]\rho \neq 0 \qquad (9)$$

$$(\texttt{while}\ e\ \texttt{do}\ c \cdot C, \rho) \to (C, \rho) \quad \text{si } [\![e]\!]\rho = 0 \qquad (10)$$

$$(\texttt{input}\ x \cdot C, \rho) \to (C, \rho[x \mapsto n]) \qquad (n \in \mathbb{Z}) \qquad (11)$$

Une configuration $(\texttt{input}\ x \cdot C, \rho)$ a donc une infinité de configurations successeurs $(C, \rho[x \mapsto n])$. Le langage n'est *pas* déterministe.

La sémantique $[\![e]\!]\rho$ des expressions est comme dans le cours. On n'aura pas besoin de connaître sa définition.

On définit aussi une sémantique dénotationnelle de CIMP, comme suit. Contrairement à celle de IMP, $[\![c]\!]\rho$ y sera maintenant un élément de $\mathcal{H}Env$, et non de $Env_\perp$. Autrement dit, pour toute commande c, $[\![c]\!]$ sera une fonction de Env vers $\mathcal{H}Env$.

$$[\![x := e]\!]\rho \stackrel{\text{def}}{=} \eta(\rho[x \mapsto [\![e]\!]\rho]) \tag{12}$$

$$[\![\texttt{skip}]\!]\rho \stackrel{\text{def}}{=} \eta(\rho) \tag{13}$$

$$[\![c_1; c_2]\!]\rho \stackrel{\text{def}}{=} [\![c_2]\!]^\dagger([\![c_1]\!]\rho) \tag{14}$$

$$[\![\texttt{if}\, e\, \texttt{then}\, c_1\ \texttt{else}\ c_2]\!]\rho \stackrel{\text{def}}{=} \begin{cases} [\![c_1]\!]\rho & \text{si } [\![e]\!]\rho \neq 0 \\ [\![c_2]\!]\rho & \text{si } [\![e]\!]\rho = 0 \end{cases} \tag{15}$$

$$[\![\texttt{while}\, e\, \texttt{do}\, c]\!]\rho \stackrel{\text{def}}{=} \text{lfp}(F_{e,c})(\rho) \tag{16}$$

$$[\![\texttt{input}\, x]\!]\rho \stackrel{\text{def}}{=} \{\rho[x \mapsto n] \mid n \in \mathbb{Z}\}, \tag{17}$$

où $F_{e,c} \colon [Env \to \mathcal{H}Env] \to [Env \to \mathcal{H}Env]$ est la fonctionnelle suivante :

$$F_{e,c}(f)(\rho) \stackrel{\text{def}}{=} \begin{cases} \eta(\rho) & \text{si } [\![e]\!]\rho = 0 \\ f^\dagger([\![c]\!]\rho) & \text{si } [\![e]\!]\rho \neq 0 \end{cases} \tag{18}$$

On rappelle que lfp signifie « plus petit point fixe ».

Env est l'ensemble de tous les environnements, ordonnés par l'égalité. En particulier :

(I) $\eta(\rho) = {\downarrow}\rho$ vaut en fait simplement $\{\rho\}$, pour tout $\rho \in Env$;

(II) tout sous-ensemble de Env est Scott-fermé, donc $\mathcal{H}Env$ est juste l'ensemble $\mathbb{P}(Env)$ des parties de Env, ordonné par inclusion ; en particulier, le côté droit $\{\rho[x \mapsto n] \mid n \in \mathbb{Z}\}$ de (17) est bien dans $\mathcal{H}Env$;

(III) tout sous-ensemble de Env est Scott-fermé, donc $\overline{A} = A$ pour tout sous-ensemble A de Env ; en particulier, $f^\dagger(F)$ vaut simplement $\bigcup_{x \in F} f(x)$, et pour toute famille dirigée $(F_i)_{i \in I}$ de $\mathcal{H}Env$, $\sup_{i \in I} F_i = \bigcup_{i \in I} F_i$;

(IV) toute fonction de Env vers $\mathcal{H}Env$ est continue : donc $[Env \to \mathcal{H}Env]$ est juste le dcpo des fonctions de Env vers $\mathcal{H}Env$, ordonné point à point.

On notera que $F_{e,c}$ est Scott-continue, pour toute expression e et toute commande c. C'est une conséquence immédiate de la **Question 3**, qu'il n'y a pas besoin de démontrer.

Question 4 Montrer que l'expression $\mathrm{lfp}(F_{e,c})(\rho)$ de la ligne (16) est bien définie, et est égale à $\sup_{n\in\mathbb{N}} F^n_{e,c}(bot)(\rho)$, pour une certaine fonction $bot \in [Env \to \mathcal{H}Env]$ que vous préciserez. Citez le théorème que vous utilisez, et vérifiez que ses hypothèses sont vérifiées.

On peut démontrer le résultat de correction suivant, et je vais vous demander de compléter certains points de sa démonstration. ($C^;$ est défini comme dans le cours : $\epsilon^; \stackrel{\text{def}}{=} \texttt{skip}$, $(c \cdot C)^; \stackrel{\text{def}}{=} c; C^;$.)

Proposition 1 *Pour toutes configurations (C, ρ) et (C', ρ'), si $(C, \rho) \to (C', \rho')$ alors $[\![C'^;]\!]\rho' \subseteq [\![C^;]\!]\rho$.*

Démonstration. Par analyse de cas sur la règle utilisée. Dans le cas des règles (4), (5), (7), (8) ainsi que (10), il est facile de voir qu'on a en fait égalité.

Dans le cas de la règle (6), on a aussi égalité. En effet, en réutilisant les notations de cette règle, on a $[\![(c_1 \cdot c_2 \cdot C)^;]\!]\rho = ([\![C^;]\!]^\dagger \circ [\![c_2]\!])^\dagger([\![c_1]\!]\rho))$ et $[\![(c_1; c_2 \cdot C)^;]\!]\rho = [\![C^;]\!]^\dagger([\![c_2]\!]^\dagger([\![c_1]\!]\rho)$, or $([\![C^;]\!]^\dagger \circ [\![c_2]\!])^\dagger = [\![C^;]\!]^\dagger \circ [\![c_2]\!]^\dagger$ par (3).

Dans le cas de la règle (9), on a aussi égalité : sous l'hypothèse $[\![e]\!]\rho \neq 0$, on montre que $[\![C^;]\!]^\dagger(\mathrm{lfp}(F_{e,c})(\rho)) = ([\![C^;]\!]^\dagger \circ \mathrm{lfp}(F_{e,c}))^\dagger([\![c]\!]\rho)$. Pour ceci, en utilisant (3) comme ci-dessus, le problème se ramène à :

Question 5 montrer que $f(\rho) = f^\dagger([\![c]\!]\rho)$, où $f \stackrel{\text{def}}{=} \mathrm{lfp}(F_{e,c})$, sous l'hypothèse $[\![e]\!]\rho \neq 0$.

Dans le cas de la règle (11), on doit :

Question 6 montrer que $[\![C^;]\!](\rho[x \mapsto n]) \subseteq [\![C^;]\!]^\dagger(\{\rho[x \mapsto m] \mid m \in \mathbb{Z}\})$.

Ceci conclut la preuve de la proposition. □

Question 7 En déduire le théorème de correction : si $(C, \rho) \to^* (\epsilon, \rho_\infty)$, alors $\rho_\infty \in [\![C^;]\!]\rho$.

Passons à l'adéquation : nous allons montrer que $[\![c]\!]\rho$ est exactement l'ensemble des environnements ρ_∞ que l'on peut obtenir à la fin d'un calcul $(c \cdot \epsilon, \rho) \to^* (\epsilon, \rho_\infty)$ quelconque partant de la configuration fixée $(c \cdot \epsilon, \rho)$.

Question 8 Montrer que, pour tous environnements $\rho, \rho_\infty \in Env$ tels que $\rho_\infty \in [\![\texttt{while}\, e\, \texttt{do}\, c]\!]\rho$, il existe un entier $n \in \mathbb{N}$ tel que $\rho_\infty \in F^n_{e,c}(bot)(\rho)$, où *bot* est la fonction découverte en **Question 4**.

Disons qu'une commande c est *adéquate* si et seulement si, pour tous $\rho, \rho' \in Env$ tels que $\rho' \in [\![c]\!]\rho$, pour tout contexte C, il existe une trace $(c \cdot C, \rho) \to^* (C, \rho')$. (Une trace est une suite finie d'étapes de calcul, comme dans le cours.)

Par exemple, toute affectation $x := e$ est adéquate : si $\rho' \in [\![x := e]\!]\rho = \{\rho[x \mapsto [\![e]\!]\rho]\}$ (par (I)), alors $\rho' = \rho[x \mapsto [\![e]\!]\rho]$, et l'on peut donc former la trace $(x := e \cdot C, \rho) \to (C, \rho')$ par la règle (4). De même, `skip` est adéquate.

Question 9 Montrer que `input` x est adéquate. Précisez la règle utilisée.

Question 10 Montrer que si c est adéquate, alors `while` e `do` c est adéquate, pour n'importe quelle expression e. C'est l'autre question difficile de l'examen.

De même, on peut démontrer que si c_1 et c_2 sont adéquates, alors $c_1; c_2$ et `if` e `then` c_1 `else` c_2 sont adéquates. Par récurrence sur la commande c, on en déduit que tout commande c est adéquate. En appliquant la définition de « adéquat » au contexte $C \stackrel{\text{def}}{=} \epsilon$, on obtient que pour tous $\rho, \rho_\infty \in Env$ tels que $\rho_\infty \in [\![c]\!]\rho$, il existe une trace $(c \cdot \epsilon, \rho) \to^* (\epsilon, \rho_\infty)$. Grâce à la **Question 7**, on en déduit que $[\![c]\!]\rho$ est *exactement* l'ensemble des environnements ρ_∞ que l'on peut atteindre comme extrémité des traces $(c \cdot \epsilon, \rho) \to^* (\epsilon, \rho_\infty)$ commençant en $(c \cdot \epsilon, \rho)$.

A Une démonstration évidente mais fausse de $f[\overline{A}] \subseteq \overline{f[A]}$

On pourrait penser procéder comme suit pour démontrer que $f[\overline{A}] \subseteq \overline{f[A]}$, lorsque f est Scott-continue : les éléments de $\overline{A}$ sont les sups de familles dirigées d'éléments de A, et f est Scott-continue, donc tout élément de $f[\overline{A}]$ est un sup (dirigé) d'éléments de $f[A]$.

Cet argument, malheureusement, est complètement faux. Les éléments de $\overline{A}$ ne sont *pas* en général les sups de familles dirigées d'éléments de A.

Voici un contre-exemple. On définit X comme l'ensemble des couples (i, x) de $[0, 1]^2$ ordonnés par $(i, x) \leq (j, y)$ si et seulement si $i = j$ et $x \leq y$, ou bien $i \leq j$ et $y = 1$. On peut vérifier que X est un dcpo. Le sous-ensemble A formé des couples (i, x) tels que $i < 1$ et $x < 1$ est tel que le point $(1, 1)$ n'est pas un sup d'aucune famille dirigée d'éléments de A, et pourtant il est dans $\overline{A}$. En effet, $(1, 1)$ est le sup (dirigé) des points $(i, 1)$ avec $i < 1$, et chaque point $(i, 1)$ est le sup (dirigé) des points (i, x) avec $x < 1$, et ces derniers sont dans A. Donc tout fermé contenant A doit contenir les points $(i, 1)$ avec $i < 1$, et de là le point $(1, 1)$. Tout fermé étant clos par le bas, tout fermé contenant A doit aussi contenir tous les points inférieurs ou égaux à $(1, 1)$, c'est-à-dire tous les points de X : on a en fait $\overline{A} = X$.