

# 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 souviens 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,  $\overline{A}$ .  $\square$

**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.

*Comme toute intersection de fermés de Scott est un fermé de Scott,  $\mathcal{H}X$  est un treillis complet, donc un dcpo. Par définition, le sup d'une famille (pas forcément dirigée)  $(F_i)_{i \in I}$  est le plus petit de ses majorants, c'est-à-dire le plus petit fermé qui contient  $\bigcup_{i \in I} F_i$ , c'est-à-dire l'adhérence de  $\bigcup_{i \in I} F_i$ .*

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

*Si  $x \leq y$  et  $y \in f^{-1}(F)$ , alors  $f(x) \leq f(y)$  car  $f$  est monotone, et  $f(y)$  est dans  $F$ , donc  $f(x)$  est dans  $F$  car  $F$  est clos par le bas. Pour toute famille dirigée  $(x_i)_{i \in I}$  d'éléments de  $f^{-1}(F)$ ,  $(f(x_i))_{i \in I}$  est une famille dirigée (car  $f$  est monotone) d'éléments de  $F$ . Donc  $\sup_{i \in I} f(x_i)$  est dans  $F$ . Comme  $f$  est Scott-continue,  $f(\sup_{i \in I} x_i)$  est dans  $F$ , donc  $\sup_{i \in I} x_i$  est dans  $f^{-1}(F)$ .*

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

**Lemme 2** *Pour toute fonction Scott-continue  $f: X \rightarrow Y$  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 fautive : voir l'annexe A.)  $\square$

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.)  $\square$

**Question 3** Pour tous dcpos  $X$  et  $Y$ , et pour toute fonction Scott-continue  $f: X \rightarrow \mathcal{H}Y$ , 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).

*D'abord,  $f^\dagger$  est monotone : si  $F \subseteq F'$ , alors  $\bigcup_{x \in F} f(x) \subseteq \bigcup_{x \in F'} f(x)$ , et donc  $\overline{\bigcup_{x \in F} f(x)} \subseteq \overline{\bigcup_{x \in F'} f(x)}$  par le lemme 1.*

*Pour toute famille dirigée  $(F_i)_{i \in I}$  dans  $\mathcal{H}X$ , on a donc  $f^\dagger(\sup_{i \in I} F_i) \supseteq \sup_{i \in I} f^\dagger(F_i)$ , par monotonie.*

*Pour démontrer l'inclusion inverse, soit  $F \stackrel{\text{def}}{=} \sup_{i \in I} f^\dagger(F_i) = \overline{\bigcup_{i \in I} f^\dagger(F_i)}$ . On doit démontrer que  $f^\dagger(\sup_{i \in I} F_i) \subseteq F$ . Comme  $F$  est fermé et que  $f^\dagger(\sup_{i \in I} F_i)$  est le plus petit fermé contenant  $\bigcup_{x \in \sup_{i \in I} F_i} f(x)$ , il suffit de démontrer que pour tout  $x \in \sup_{i \in I} F_i$ ,  $f(x) \subseteq F$ , c'est-à-dire que  $f[\overline{A}]$  est inclus dans l'ensemble  $\downarrow F$  des fermés inclus dans  $F$ , où  $A \stackrel{\text{def}}{=} \bigcup_{i \in I} F_i$  (donc  $\sup_{i \in I} F_i = \overline{A}$ ). Par le lemme 2, on a  $f[\overline{A}] \subseteq \overline{f[A]}$ . Il reste donc à démontrer que  $\overline{f[A]} \subseteq \downarrow F$ . Comme  $\downarrow F$  est un fermé de Scott de  $\mathcal{H}X$ , par le lemme 3, il suffit de démontrer que  $f[A] \subseteq \downarrow F$ , c'est-à-dire que pour tout  $x \in A$ ,  $f(x)$  est inclus dans  $F$ . Or pour tout  $x \in A$ ,  $x$  est dans un  $F_i$ , donc  $f(x) \in f^\dagger(F_i) \subseteq F$ .*

On dispose aussi d'une fonction  $\eta: X \rightarrow \mathcal{H}X$  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 lemme 3. De plus, on admettra que  $\eta$  est Scott-continue. Dans la suite, on pourra utiliser sans les redémontrer les égalités suivantes :

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

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

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

## 2 Le langage CImp

Considérons le langage CIMP dont la syntaxe est la suivante :

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

$$(x := e \cdot C, \rho) \rightarrow (C, \rho[x \mapsto \llbracket e \rrbracket \rho]) \quad (4)$$

$$(\text{skip} \cdot C, \rho) \rightarrow (C, \rho) \quad (5)$$

$$(c_1; c_2 \cdot C, \rho) \rightarrow (c_1 \cdot c_2 \cdot C, \rho) \quad (6)$$

$$(\text{if } e \text{ then } c_1 \text{ else } c_2 \cdot C, \rho) \rightarrow (c_1 \cdot C, \rho) \text{ si } \llbracket e \rrbracket \rho \neq 0 \quad (7)$$

$$(\text{if } e \text{ then } c_1 \text{ else } c_2 \cdot C, \rho) \rightarrow (c_2 \cdot C, \rho) \text{ si } \llbracket e \rrbracket \rho = 0 \quad (8)$$

$$(\text{while } e \text{ do } c \cdot C, \rho) \rightarrow (c \cdot \text{while } e \text{ do } c \cdot C, \rho) \text{ si } \llbracket e \rrbracket \rho \neq 0 \quad (9)$$

$$(\text{while } e \text{ do } c \cdot C, \rho) \rightarrow (C, \rho) \text{ si } \llbracket e \rrbracket \rho = 0 \quad (10)$$

$$(\text{input } x \cdot C, \rho) \rightarrow (C, \rho[x \mapsto n]) \quad (n \in \mathbb{Z}) \quad (11)$$

Une configuration (**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  $\llbracket e \rrbracket \rho$  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,  $\llbracket c \rrbracket \rho$  y sera maintenant un élément de  $\mathcal{HEnv}$ , et non de  $Env_{\perp}$ . Autrement dit, pour toute commande  $c$ ,  $\llbracket c \rrbracket$  sera une fonc-

tion de  $Env$  vers  $\mathcal{H}Env$ .

$$\llbracket x := e \rrbracket \rho \stackrel{\text{def}}{=} \eta(\rho[x \mapsto \llbracket e \rrbracket \rho]) \quad (12)$$

$$\llbracket \text{skip} \rrbracket \rho \stackrel{\text{def}}{=} \eta(\rho) \quad (13)$$

$$\llbracket c_1; c_2 \rrbracket \rho \stackrel{\text{def}}{=} \llbracket c_2 \rrbracket^\dagger(\llbracket c_1 \rrbracket \rho) \quad (14)$$

$$\llbracket \text{if } e \text{ then } c_1 \text{ else } c_2 \rrbracket \rho \stackrel{\text{def}}{=} \begin{cases} \llbracket c_1 \rrbracket \rho & \text{si } \llbracket e \rrbracket \rho \neq 0 \\ \llbracket c_2 \rrbracket \rho & \text{si } \llbracket e \rrbracket \rho = 0 \end{cases} \quad (15)$$

$$\llbracket \text{while } e \text{ do } c \rrbracket \rho \stackrel{\text{def}}{=} \text{lfp}(F_{e,c})(\rho) \quad (16)$$

$$\llbracket \text{input } x \rrbracket \rho \stackrel{\text{def}}{=} \{\rho[x \mapsto n] \mid n \in \mathbb{Z}\}, \quad (17)$$

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

$$F_{e,c}(f)(\rho) \stackrel{\text{def}}{=} \begin{cases} \eta(\rho) & \text{si } \llbracket e \rrbracket \rho = 0 \\ f^\dagger(\llbracket c \rrbracket \rho) & \text{si } \llbracket e \rrbracket \rho \neq 0 \end{cases} \quad (18)$$

On rappelle que  $\text{lfp}$  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  $\bar{A} = A$  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  $[Env \rightarrow \mathcal{H}Env]$  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  $\text{lfp}(F_{e,c})(\rho)$  de la ligne (16) est bien définie, et est égale à  $\sup_{n \in \mathbb{N}} F_{e,c}^n(\text{bot})(\rho)$ , pour une certaine fonction  $\text{bot} \in [Env \rightarrow \mathcal{H}Env]$  que vous préciserez. Citez le théorème que vous utilisez, et vérifiez que ses hypothèses sont vérifiées.

C'est le théorème de point fixe de Kleene.  $[Env \rightarrow HEnv]$  est un dcpo, comme l'est tout ensemble de fonctions Scott-continues d'un dcpo vers un autre. De plus, il est pointé, car la fonction  $bot$  identiquement égale à  $\emptyset$  est son plus petit élément.

Ce théorème énonce qu'alors  $\text{lfp}(F_{e,c}) = \sup_{n \in \mathbb{N}} F_{e,c}^n(bot)$ . Les sups dirigés dans les espaces de fonctions étant calculés point à point, on a  $\text{lfp}(F_{e,c})(\rho) = \sup_{n \in \mathbb{N}} (F_{e,c}^n(bot)(\rho))$ .

A noter que le théorème de point fixe de Tarski ne s'appliquerait que partiellement.  $HEnv$  est un treillis complet, et donc  $[Env \rightarrow HEnv]$  aussi. Mais Tarski ne donnerait pas la formule demandée  $\sup_{n \in \mathbb{N}} F_{e,c}^n(bot)(\rho)$ .

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 :  $e^; \stackrel{\text{def}}{=} \text{skip}$ ,  $(c \cdot C)^; \stackrel{\text{def}}{=} c; C^;$ .)

**Proposition 1** Pour toutes configurations  $(C, \rho)$  et  $(C', \rho')$ , si  $(C, \rho) \rightarrow (C', \rho')$  alors  $\llbracket C'^; \rrbracket \rho' \subseteq \llbracket C^; \rrbracket \rho$ .

**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  $\llbracket (c_1 \cdot c_2 \cdot C)^; \rrbracket \rho = (\llbracket C^; \rrbracket^\dagger \circ \llbracket c_2 \rrbracket)^\dagger(\llbracket c_1 \rrbracket \rho)$  et  $\llbracket (c_1; c_2 \cdot C)^; \rrbracket \rho = \llbracket C^; \rrbracket^\dagger(\llbracket c_2 \rrbracket^\dagger(\llbracket c_1 \rrbracket \rho))$ , or  $(\llbracket C^; \rrbracket^\dagger \circ \llbracket c_2 \rrbracket)^\dagger = \llbracket C^; \rrbracket^\dagger \circ \llbracket c_2 \rrbracket^\dagger$  par (3).

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

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

Par définition,  $f$  est un point fixe de  $F_{e,c}$ . Donc  $f(\rho) = F_{e,c}(f)(\rho)$ , et ceci est égal à  $f^\dagger(\llbracket c \rrbracket \rho)$  puisque  $\llbracket e \rrbracket \rho \neq 0$ .

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

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

Le côté droit,  $\llbracket C^; \rrbracket^\dagger(\{\rho[x \mapsto m] \mid m \in \mathbb{Z}\})$ , vaut (par définition de l'opération  $f \mapsto f^\dagger$ ) [l'adhérence de] l'union des ensembles  $\llbracket C^; \rrbracket(\rho[x \mapsto m])$  lorsque  $m$  parcourt  $\mathbb{Z}$ , et contient donc celui que

l'on obtient en posant  $m \stackrel{\text{def}}{=} n$ , qui est le côté gauche  $\llbracket C \rrbracket(\rho[x \mapsto n])$  de l'inclusion demandée.

Ceci conclut la preuve de la proposition.  $\square$

**Question 7** En déduire le théorème de correction : si  $(C, \rho) \rightarrow^* (\epsilon, \rho_\infty)$ , alors  $\rho_\infty \in \llbracket C \rrbracket \rho$ .

Par récurrence sur la longueur de la trace de  $(C, \rho)$  à  $(\epsilon, \rho_\infty)$ .  
 Si elle vaut 0, alors  $C = \epsilon$ ,  $\llbracket C \rrbracket \rho = \eta(\rho) = \{\rho\}$  par (I), et  $\rho_\infty = \rho$  est bien dans cet ensemble.  
 Sinon, on a  $(C, \rho) \rightarrow (C', \rho')$  et  $\rho_\infty \in \llbracket C' \rrbracket \rho'$ , par hypothèse de récurrence. Par la proposition 1,  $\llbracket C' \rrbracket \rho' \subseteq \llbracket C \rrbracket \rho$ , donc  $\rho_\infty$  est aussi dans  $\llbracket C \rrbracket \rho$ .

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

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

Par la **Question 4**,  $\rho_\infty$  est dans  $\sup_{n \in \mathbb{N}} F_{e,c}^n(\text{bot})(\rho)$ .  
 Il serait faux d'annoncer que le sup est atteint en un  $n \in \mathbb{N}$ . Par exemple, les ensembles  $A_n \stackrel{\text{def}}{=} \{\rho[x \mapsto 0], \dots, \rho[x \mapsto n]\}$  forment une suite strictement croissante de fermés de Scott, dont le sup est l'union, qui n'est pas atteinte.  
 Mais, par (III),  $\sup_{n \in \mathbb{N}} F_{e,c}^n(\text{bot})(\rho) = \bigcup_{n \in \mathbb{N}} F_{e,c}^n(\text{bot})(\rho)$ . Comme  $\rho_\infty$  en est un élément, il appartient à  $F_{e,c}^n(\text{bot})(\rho)$  pour au moins un  $n \in \mathbb{N}$ .

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

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

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

On montre que si  $\rho' \in \llbracket \text{input } x \rrbracket \rho$ , alors pour tout contexte  $C$ , il existe une trace  $(\text{input } x \cdot C, \rho) \rightarrow^* (C, \rho')$ . Par hypothèse,  $\rho' \in \{\rho[x \mapsto n] \mid n \in \mathbb{Z}\}$ , donc il existe un entier  $n \in \mathbb{Z}$  tel que  $\rho' = \rho[x \mapsto n]$ . On utilise la règle (11) et on obtient  $(\text{input } x \cdot C, \rho) \rightarrow (C, \rho[x \mapsto n])$ .

**Question 10** Montrer que si  $c$  est adéquate, alors  $\text{while } e \text{ do } c$  est adéquate, pour n'importe quelle expression  $e$ . C'est l'autre question difficile de l'examen.

On montre que si  $\rho' \in F_{e,c}^n(\text{bot})(\rho)$ , alors il existe une trace  $(\text{while } e \text{ do } c \cdot C, \rho) \rightarrow^* (C, \rho')$ , par récurrence sur  $n \in \mathbb{N}$ , pour n'importe quel contexte  $C$ . Le résultat s'en déduira car par la **Question 8**, si  $\rho' \in \llbracket \text{while } e \text{ do } c \rrbracket \rho$ , alors  $\rho' \in F_{e,c}^n(\text{bot})(\rho)$  pour un certain  $n \in \mathbb{N}$ .

Si  $n = 0$ , alors  $\rho'$  est dans  $\text{bot}(\rho) = \emptyset$ . C'est impossible, et de l'impossible on peut déduire ce que l'on veut.

Sinon, on considère deux cas. Dans le premier cas,  $\llbracket e \rrbracket \rho = 0$ . Alors  $F_{e,c}^n(\text{bot})(\rho) = \eta(\rho) = \{\rho\}$  par (I), donc  $\rho' = \rho$ . On a alors  $(\text{while } e \text{ do } c \cdot C, \rho) \rightarrow (C, \rho)$  par (10).

Dans le second cas,  $\llbracket e \rrbracket \rho \neq 0$ . Alors  $F_{e,c}^n(\text{bot})(\rho) = f^\dagger(\llbracket c \rrbracket \rho)$ , avec  $f \stackrel{\text{def}}{=} F_{e,c}^{n-1}(\text{bot})$ .

Pour tout sous-ensemble  $A$  de  $\text{Env}$ ,  $f^\dagger(A) = \bigcup_{\rho_1 \in A} f(\rho_1)$  par (III). Comme  $\rho'$  est dans  $F_{e,c}^n(\text{bot})(\rho) = f^\dagger(\llbracket c \rrbracket \rho)$ , il existe un  $\rho_1 \in \llbracket c \rrbracket \rho$  tel que  $\rho' \in f(\rho_1)$ .

De  $\rho' \in f(\rho_1)$  on déduit  $\rho' \in F_{e,c}^{n-1}(\text{bot})(\rho_1)$ , par définition de  $f$ , donc il existe une trace  $(\text{while } e \text{ do } c \cdot C, \rho_1) \rightarrow^* (C, \rho')$  par hypothèse de récurrence.

De  $\rho_1 \in \llbracket c \rrbracket \rho$ , on déduit qu'il existe une trace  $(c \cdot \text{while } e \text{ do } c \cdot C, \rho) \rightarrow^* (\text{while } e \text{ do } c \cdot C, \rho_1)$ , par adéquation de  $c$ . (On notera qu'on utilise la propriété d'adéquation de  $c$  sur le contexte  $\text{while } e \text{ do } c \cdot C$ , pas sur  $C$ .)

En concaténant les deux traces ainsi qu'une instance de (9), on obtient  $(\text{while } e \text{ do } c \cdot C, \rho) \rightarrow^* (C, \rho')$ , et l'on conclut.

De même, on peut démontrer que si  $c_1$  et  $c_2$  sont adéquates, alors  $c_1; c_2$  et  $\text{if } e \text{ then } c_1 \text{ else } c_2$  sont adéquates. Par récurrence sur la commande  $c$ , on en déduit que toute commande  $c$  est adéquate. En appliquant la définition de « adéquat » au contexte  $C \stackrel{\text{def}}{=} \epsilon$ , on obtient que pour tous  $\rho, \rho_\infty \in \text{Env}$  tels

que  $\rho_\infty \in \llbracket c \rrbracket \rho$ , il existe une trace  $(c \cdot \epsilon, \rho) \rightarrow^* (\epsilon, \rho_\infty)$ . Grâce à la **Question 7**, on en déduit que  $\llbracket c \rrbracket \rho$  est *exactement* l'ensemble des environnements  $\rho_\infty$  que l'on peut atteindre comme extrémité des traces  $(c \cdot \epsilon, \rho) \rightarrow^* (\epsilon, \rho_\infty)$  commençant en  $(c \cdot \epsilon, \rho)$ .

## A Une démonstration évidente mais fautive 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  $\overline{A} = X$ .