Corrigé de la composition d'informatique-mathématiques ENS - MP - 2020

Corrigé

Stéphane Legros, Yann Salmon

26 février 2021

Remarque : pour alléger les notations, nous nous permettrons quelquefois de noter $\leq$ (resp. $\sup_{i \in \mathbb{N}}$) au lieu de $\leq_{\tau}$ (resp. $\sup_{i \in \mathbb{N}}^A$). D'autre part, comme dans l'exemple du bas de la page 4, il faut comprendre que le type $\tau_1 \Rightarrow \tau_2 \Rightarrow \tau_3$ est en fait $\tau_1 \Rightarrow (\tau_2 \Rightarrow \tau_3)$ (convention usuelle en Ocaml).

1 Partie I

Question 1.1. a. Les suites croissantes de $(\llbracket \text{nat} \rrbracket, \leq_{\text{nat}})$ sont la suite constante $(\perp_{\text{nat}})_{i \in \mathbb{N}}$ et les suites $(a_i)_{i \in \mathbb{N}}$ telles qu'il existe $i_0 \in \mathbb{N}$ et $k \in \mathbb{N}$ avec $a_i = \perp_{\text{nat}}$ pour $i < i_0$ et $a_i = k$ pour $i \geq i_0$.

b. La suite constante $(\perp_{\text{nat}})_{i \in \mathbb{N}}$ admet $\perp_{\text{nat}}$ pour borne supérieure et une suite croissante de la seconde forme admet k pour borne supérieure : $(\llbracket \text{nat} \rrbracket, \leq_{\text{nat}})$ est un ensemble ordonné complet.

Question 1.2. a. Soit $f : \llbracket \text{nat} \rrbracket \rightarrow \llbracket \text{unit} \rrbracket$ une application croissante et $(a_i)_{i \in \mathbb{N}}$ une suite croissante de $(\llbracket \text{nat} \rrbracket, \leq_{\text{nat}})$. Les suites $(a_i)_{i \in \mathbb{N}}$ et $(f(a_i))_{i \in \mathbb{N}}$ sont croissantes et stationnaires (resp. en k et $f(k)$ avec $k \in \llbracket \text{nat} \rrbracket$) : leurs bornes supérieures sont les valeurs auxquelles elles stationnent, donc $f(\sup_{i \in \mathbb{N}} a_i) = f(k) = \sup_{i \in \mathbb{N}} f(a_i)$ et f est continue.

b. Soit $f \in \llbracket \text{nat} \Rightarrow \text{unit} \rrbracket$ tel que $f(\perp_{\text{nat}}) \neq \perp_{\text{unit}}$. On a donc $f(\perp_{\text{nat}}) = \text{uu}$, puis

$$\forall k \in \mathbb{N}, \text{uu} = f(\perp_{\text{nat}}) \leq f(k)$$

par croissance de f : ceci impose $f(k) = \text{uu}$ et f est constante. Il y a donc une unique telle fonction, car la fonction constante convient bien.

c. Pour $i \in \mathbb{N}$, on pose $f_i : k \mapsto \begin{cases} \text{uu} & \text{si } k \in \{0, 1, \dots, i\} \\ \perp_{\text{unit}} & \text{sinon} \end{cases}$.

Question 1.3. a. Soient $a, b \in \llbracket \tau \rrbracket$ avec $a \leq b$. On a $f_i(a) \leq f_i(b)$ pour tout $i \in \mathbb{N}$, donc $f'(a) = \sup_{i \in \mathbb{N}} f_i(a) \leq \sup_{i \in \mathbb{N}} f_i(b) = f'(b)$. Ainsi f' est croissante.

Soit $(e_n)_{n \in \mathbb{N}}$ une suite croissante de $\llbracket \tau_1 \rrbracket$ et $e = \sup_{n \in \mathbb{N}} e_n$. Nous avons $e_n \leq e$ pour tout $n \in \mathbb{N}$, donc, par croissance de f' , $f'(e_n) \leq f'(e)$ pour tout $n \in \mathbb{N}$, puis $\sup_{n \in \mathbb{N}} f'(e_n) \leq f'(e)$.

Soit $\alpha \in \llbracket \tau_2 \rrbracket$ un majorant de $(f'(e_n))_{n \in \mathbb{N}}$, on a :

$$\forall i \in \mathbb{N}, \forall n \in \mathbb{N}, f_i(e_n) \leq \alpha.$$

On en déduit, par continuité des f_i :

$$\forall i \in \mathbb{N}, f_i(e) = f_i(\sup_{n \in \mathbb{N}} e_n) = \sup_{n \in \mathbb{N}} f_i(e_n) \leq \alpha$$

et donc $f'(e) \leq \alpha$. Ainsi $f'(e)$ est le plus petit des majorants de la suite $(f'(e_n))_{n \in \mathbb{N}}$ et f' est continue.

- b. Soit $(f_i)_{i \in \mathbb{N}}$ une suite croissante de $\llbracket \tau_1 \Rightarrow \tau_2 \rrbracket$ (pour l'ordre $\leq_{\tau_1 \Rightarrow \tau_2}$) et soit f' la fonction définie à la question précédente (qui établit que $f' \in \llbracket \tau_1 \Rightarrow \tau_2 \rrbracket$). On a :

$$\forall i \in \mathbb{N}, \forall e \in \llbracket \tau_1 \rrbracket, f_i(e) \leq_{\tau_2} f'(e)$$

donc f' est un majorant de $(f_i)_{i \in \mathbb{N}}$ pour l'ordre $\leq_{\tau_1 \Rightarrow \tau_2}$.

Si $g \in \llbracket \tau_1 \Rightarrow \tau_2 \rrbracket$ majore la suite $(f_i)_{i \in \mathbb{N}}$, nous avons :

$$\forall i \in \mathbb{N}, \forall e \in \llbracket \tau_1 \rrbracket, f_i(e) \leq_{\tau_2} g(e)$$

et donc

$$\forall e \in \llbracket \tau_1 \rrbracket, f'(e) = \sup_{i \in \mathbb{N}} f_i(e) \leq_{\tau_2} g(e)$$

donc $f' \leq_{\tau_1 \Rightarrow \tau_2} g$: f' est la borne supérieure de $(f_i)_{i \in \mathbb{N}}$ et $(\llbracket \tau_1 \Rightarrow \tau_2 \rrbracket, \leq_{\tau_1 \Rightarrow \tau_2})$ est un ordre partiel complet.

2 Partie II

Question 2.1. Le programme proposé est bien typé à la condition que $t(x) = t(y) = t(z)$ et que ce type soit dans $\{\text{unit}, \text{bool}, \text{nat}\}$; il est en ce cas de type $t(x) \Rightarrow t(x)$.

Question 2.2. a. Il suffit d'utiliser les définitions; on a successivement, pour tout $n \in \mathbb{N}$:

- $\llbracket x \rrbracket^{\mathcal{E}[x \mapsto n]} = n$;
- $\llbracket x/2 \rrbracket^{\mathcal{E}[x \mapsto n]} = \lfloor n/2 \rfloor$ et $\llbracket 2 \rrbracket^{\mathcal{E}[x \mapsto n]} = 2$;
- $\llbracket 2 \times (x/2) \rrbracket^{\mathcal{E}[x \mapsto n]} = 2 \lfloor n/2 \rfloor$;
- $\llbracket (2 \times (x/2)) = x \rrbracket^{\mathcal{E}[x \mapsto n]} = \begin{cases} \text{tt} & \text{si } 2 \lfloor n/2 \rfloor = n \\ \text{ff} & \text{si } 2 \lfloor n/2 \rfloor \neq n \end{cases}$

et donc

$$\llbracket p_e \rrbracket^{\mathcal{E}}(n) = \llbracket (2 \times (x/2)) = x \rrbracket^{\mathcal{E}[x \mapsto n]} = \begin{cases} \text{tt} & \text{si } n \text{ est pair} \\ \text{ff} & \text{si } n \text{ est impair} \end{cases}.$$

L'ensemble cherché est donc l'ensemble des entiers naturels pairs.

- b. On propose $p_c = \text{fun } x \rightarrow (\text{if } (p_e \ x) \text{ then } x/2 \text{ else } ((3 \times x) + 1))$.

Question 2.3. a. On a

$$\llbracket p \rrbracket(f)(e) = \begin{cases} \perp_{\text{nat}} & \text{si } e = \perp_{\text{nat}} \\ 1 & \text{si } e = 0 \\ \perp_{\text{nat}} & \text{si } e \in \mathbb{N}^* \text{ et } f(e-1) = \perp_{\text{nat}} \\ e \cdot f(e-1) & \text{sinon} \end{cases}.$$

- b. Montrons par récurrence que pour tout $i \in \mathbb{N}$,

$$\forall e \in \llbracket \text{nat} \rrbracket, \llbracket p \rrbracket^{i+1}(\perp_{\text{nat} \Rightarrow \text{nat}})(e) = \begin{cases} \perp_{\text{nat}} & \text{si } e = \perp_{\text{nat}} \\ e! & \text{si } e \in \{0, 1, \dots, i\} \\ \perp_{\text{nat}} & \text{sinon} \end{cases}.$$

Pour $i = 0$, la propriété est vérifiée car $\perp_{\text{nat} \Rightarrow \text{nat}}(e-1) = \perp_{\text{nat}}$ pour $e \geq 1$. Soit $i \in \mathbb{N}$ et supposons que la relation soit vraie au rang i . Soit $e \in \llbracket \text{nat} \rrbracket$: $\llbracket p \rrbracket^{i+2}(\perp)(e) = \llbracket p \rrbracket(\llbracket p \rrbracket^{i+1}(\perp))(e)$, donc

- si $e = \perp_{\text{nat}}$, alors $\llbracket p \rrbracket^{i+2}(\perp)(e) = \perp_{\text{nat}}$;
- si $e = 0$, $\llbracket p \rrbracket^{i+2}(\perp)(e) = 1 = e!$;
- si $e \geq i+2$, alors $e-1 \geq i+1$, donc $\llbracket p \rrbracket^{i+1}(\perp)(e-1) = \perp_{\text{nat}}$ (HR) puis $\llbracket p \rrbracket^{i+2}(\perp)(e) = \perp_{\text{nat}}$;
- sinon, $e \in \{1, \dots, i+1\}$, donc $\llbracket p \rrbracket^{i+1}(\perp)(e-1) = (e-1)! \neq \perp_{\text{nat}}$ (HR) et donc $\llbracket p \rrbracket^{i+2}(\perp)(e) = e \cdot \llbracket p \rrbracket^{i+1}(\perp)(e-1) = e!$. Récurrence établie.

- c. Pour $k \in \mathbb{N}$, la suite $(\llbracket p \rrbracket^i(\perp)(k))_{i \in \mathbb{N}}$ stationne à la valeur $k!$, donc $\llbracket \text{rec } p \rrbracket(k) = k!$ (comme démontré à la question 1.3.a, la limite d'une suite croissante (f_i) d'éléments de $\llbracket \tau_1 \Rightarrow \tau_2 \rrbracket$ est la fonction f' définie par $f'(e) = \sup_{i \in \mathbb{N}} f_i(e)$).

Comme $\llbracket p \rrbracket^{i+1}(\perp)(\perp_{\text{nat}}) = \perp_{\text{nat}}$ pour tout i , $\llbracket \text{rec } p \rrbracket(\perp_{\text{nat}}) = \perp_{\text{nat}}$.

Question 2.4. a. Le programme $q \stackrel{\text{def}}{=} \text{fun } f \rightarrow \text{fun } n \rightarrow \text{if } n = 1 \text{ then } uu \text{ else } f(f_c n)$ convient. On a en effet, pour $g \in \llbracket \text{nat} \Rightarrow \text{unit} \rrbracket$ et $e \in \llbracket \text{nat} \rrbracket$:

$$\llbracket q \rrbracket (g)(e) \begin{cases} uu & \text{si } e = 1 \\ \perp_{\text{unit}} & \text{si } e = \perp_{\text{nat}} \\ g(f_c(e)) & \text{sinon} \end{cases}$$

et la propriété demandée est alors évidente par récurrence.

b. On pose $p \stackrel{\text{def}}{=} \text{rec } q$. Pour $e \in \mathbb{N}^*$, $\llbracket q \rrbracket^i(e)$ vaut $\perp_{\text{unit}}$ si $f_c^k(e) \neq 1$ pour tout $k \leq i$ et uu sinon. La suite $(\llbracket q \rrbracket^i(e))_{i \in \mathbb{N}}$ va donc stationner à $\perp_{\text{unit}}$ si $f_c^k(e) \neq 1$ pour tout $k \in \mathbb{N}$ et stationner à uu sinon. On en déduit que $\llbracket \text{rec } q \rrbracket(e) = uu$ si et seulement s'il existe $k \in \mathbb{N}$, $f_c^k(e) = 1$.

Question 2.5. a. $\llbracket \text{unit} \Rightarrow \text{unit} \rrbracket$ contient les trois éléments :

$$g_1 : \alpha \mapsto \perp_{\text{unit}}, \quad g_2 : \alpha \mapsto \alpha, \quad g_3 : \alpha \mapsto uu,$$

et en fixant une variable x de type unit , ces trois fonctions sont calculées par les programmes

$$p_1 = \text{fun } x \rightarrow \perp_{\text{unit}}, \quad p_2 = \text{fun } x \rightarrow x, \quad p_3 = \text{fun } x \rightarrow uu.$$

b. Comme l'ensemble des programmes est dénombrable, l'ensemble $\mathcal{P}$ des programmes $p : \llbracket \text{nat} \Rightarrow \text{unit} \rrbracket$ dont l'interprétation est indépendante de l'environnement est également dénombrable; on en déduit que les éléments calculables de $\llbracket \text{nat} \Rightarrow \text{unit} \rrbracket$ sont en quantité au plus dénombrable (image de $\mathcal{P}$ par l'application $p \mapsto \llbracket p \rrbracket$). Comme l'ensemble des applications de nat dans unit est plus que dénombrable (cet ensemble est équipotent à $\mathcal{P}(\text{nat})$: il a la puissance du continu), il existe des éléments non calculables dans $\llbracket \text{nat} \Rightarrow \text{unit} \rrbracket$.

Question 2.6. Le programme $\text{fun } x \rightarrow \text{fun } y \rightarrow \text{if } x = uu \text{ then } y \text{ else } \text{DIV}_{\text{unit}}$ convient.

Question 2.7. a. Soit $f \in \llbracket \text{bool} \Rightarrow \text{unit} \rrbracket$. On a, $\llbracket p \rrbracket(f) = \llbracket (x \text{ tt}) \&_u (x \text{ ff}) \rrbracket^{[x \mapsto f]}$, ce qui vaut uu si et seulement si l'un des deux termes s'évalue en uu . L'ensemble cherché est donc celui des fonctions qui prennent la valeur uu (en tt ou en ff).

b. C'est l'ensemble des $f \in \llbracket \text{nat} \Rightarrow \text{unit} \rrbracket$ telles qu'il existe $n \in \mathbb{N}$ tel que $f(n) = uu$.

3 Partie III

Question 3.1. Si x est une variable de type τ , $\llbracket \tau \rrbracket$ et son complémentaire sont calculés respectivement par $\text{fun } x \rightarrow uu$ et $\text{fun } x \rightarrow \text{DIV}_{\text{unit}}$: $\llbracket \tau \rrbracket$ est donc un ouvert calculatoire et un fermé calculatoire.

Question 3.2. D'après la question 2.5.a, les ouverts calculatoires de $\llbracket \text{unit} \rrbracket$ sont les parties $\emptyset$, $\{uu\}$ et $\llbracket \text{unit} \rrbracket$. La quatrième partie $\{\perp_{\text{unit}}\}$ n'est pas un ouvert calculatoire car son indicatrice, faute d'être croissante, n'est pas dans $\llbracket \text{unit} \Rightarrow \text{unit} \rrbracket$.

Question 3.3. Il existe deux programmes $p : \tau \Rightarrow \tau'$ et $q : \tau' \Rightarrow \text{unit}$ tels que $f = \llbracket p \rrbracket$ et $\mathbf{1}_U = \llbracket q \rrbracket$, en notant $\mathbf{1}_U$ la fonction indicatrice d'une partie U . En fixant une variable x de type τ' , on peut définir le programme $r \stackrel{\text{def}}{=} \text{fun } x \rightarrow q(p x)$, qui est de type $\tau \Rightarrow \text{unit}$ et on a, pour tout $e \in \llbracket \tau \rrbracket$:

$$\llbracket r \rrbracket(e) = uu \text{ s.s.i } \llbracket q \rrbracket(\llbracket p \rrbracket(e)) = uu \text{ s.s.i } \llbracket p \rrbracket(e) \in U \text{ s.s.i } f(e) \in U \text{ s.s.i } e \in f^{-1}(U)$$

donc $f^{-1}(U)$ est un ouvert calculatoire.

Question 3.4. Soient U et V deux ouverts calculatoires de $\llbracket \tau \rrbracket$, $p, q : \tau \Rightarrow \text{unit}$ deux programmes tels que $\llbracket p \rrbracket = \mathbf{1}_U$ et $\llbracket q \rrbracket = \mathbf{1}_V$ et x une variable de type τ . Les programmes $\text{fun } x \rightarrow (p x \parallel_u q x)$ et $\text{fun } x \rightarrow (p x \&_u q x)$ sont alors de type $\tau \Rightarrow \text{unit}$ et calculent respectivement $U \cup V$ et $U \cap V$.

Question 3.5. On pose $q \stackrel{\text{def}}{=} \text{fun } f \rightarrow \text{fun } k \rightarrow \text{fun } e \rightarrow (p k e \parallel_u f(k+1) e)$, de type $\tau' \Rightarrow \tau'$ avec $\tau' = \text{nat} \Rightarrow \tau \Rightarrow \text{unit}$. On a, pour $k \in \mathbb{N}$ et $e \in \llbracket \tau \rrbracket$:

$$\begin{cases} \llbracket q \rrbracket(\perp)(k)(e) = uu \text{ s.s.i } \llbracket p \rrbracket(k)(e) = uu \text{ s.s.i } e \in U_k \\ \llbracket q \rrbracket^{i+1}(\perp)(k)(e) = uu \text{ s.s.i } \llbracket p \rrbracket(k)(e) = uu \text{ ou } \llbracket q \rrbracket^i(\perp)(k+1)(e) = uu \end{cases}$$

et donc par récurrence :

$$\forall i \in \mathbb{N}, \llbracket q \rrbracket^i(\perp)(k)(e) = uu \text{ s.s.i } \exists j \in \{k, k+1, \dots, k+i-1\}, e \in U_j$$

puis

$$\llbracket \text{rec } q \rrbracket(k)(e) = uu \text{ s.s.i } e \in \bigcup_{j \geq k} U_j.$$

Le programme $(\text{rec } q) 0$ calcule donc $\bigcup_{j \geq 0} U_j$, qui est ainsi un ouvert calculatoire.

Question 3.6. a. On souhaite ici que $\llbracket \forall_{\emptyset} p \rrbracket = uu$ pour tout programme $p : \tau \Rightarrow \text{unit}$, puisque la propriété $(\forall e \in \emptyset, \llbracket p \rrbracket (e) = uu)$ est vraie pour tout tel programme p . On fixe donc une variable x de type $\tau \Rightarrow \text{unit}$ et on définit $\forall_{\emptyset} \stackrel{\text{def}}{=} \text{fun } x \rightarrow uu$.

b. Si $p : \tau \Rightarrow \text{unit}$, alors $\llbracket p \rrbracket$ est croissante. Or E contient $\perp_{\tau}$, donc la propriété $\forall e \in E, \llbracket p \rrbracket (e) = uu$ se ramène à $\llbracket p \rrbracket (\perp_{\tau}) = uu$. Le sens direct est évident ; pour le sens réciproque, on a pour $e \in E, \perp_{\tau} \leq_{\tau} e$, donc $uu = \llbracket p \rrbracket (\perp_{\tau}) \leq_{\text{unit}} \llbracket p \rrbracket (e)$, c'est-à-dire $\llbracket p \rrbracket (e) = uu$.

En fixant une variable x de type $\tau \Rightarrow \text{unit}$, le programme $\forall_E \stackrel{\text{def}}{=} \text{fun } x \rightarrow x \text{ DIV}_{\tau}$ donne pour $p : \tau \Rightarrow \text{unit}$

$$\llbracket \forall_E p \rrbracket = uu \text{ s.s.i } \llbracket p \text{ DIV}_{\tau} \rrbracket = uu \text{ s.s.i } \llbracket p \rrbracket (\perp_{\tau}) = uu \text{ s.s.i } \forall e \in E, \llbracket p \rrbracket (e) = uu.$$

Toute partie contenant $\perp_{\tau}$ est donc calculatoirement compacte.

c. Soit A un fermé calculatoire de $\llbracket \tau \rrbracket$ et $p : \tau \Rightarrow \text{unit}$ un programme tel que $\mathbf{1}_{\llbracket \tau \rrbracket \setminus A} = \llbracket p \rrbracket$. Si A contient $\perp_{\tau}$, il est calculatoirement compact d'après le (b). Sinon, $\perp_{\tau} \in \llbracket \tau \rrbracket \setminus A$ et $\llbracket p \rrbracket (\perp_{\tau}) = uu$. Par croissance, de $\llbracket p \rrbracket$, on a $\llbracket p \rrbracket (e) = uu$ pour tout $e \in \llbracket \tau \rrbracket$. Par conséquent, $A = \emptyset$, qui est calculatoirement compact d'après le (a).

Question 3.7. On montre par récurrence sur son cardinal qu'une partie finie de $\mathbb{N}$ est calculatoirement compacte :

- $\emptyset$ est une partie calculatoirement compacte dans $\llbracket \text{nat} \rrbracket$ d'après la question précédente.
- soit $k \in \mathbb{N}$ et supposons que toute partie de cardinal k de $\mathbb{N}$ est calculatoirement compacte dans $\llbracket \text{nat} \rrbracket$; si A est une partie de $\mathbb{N}$ de cardinal $k + 1$, on fixe $n \in \mathbb{N}$ et B de cardinal k tels que $A = B \uplus \{n\}$. Comme B est calculatoirement compact, on peut définir un programme $\forall_B$. On pose ensuite $\forall_A = \text{fun } f \rightarrow (f \ n) \ \&_{\text{u}} \ \forall_B \ f$ et on a pour tout programme $p : \text{nat} \Rightarrow \text{unit}$:

$$\llbracket \forall_A p \rrbracket = uu \text{ s.s.i } \llbracket p \rrbracket (n) = uu \text{ et } \forall e \in B, \llbracket p \rrbracket (e) = uu \text{ s.s.i } \forall e \in A, \llbracket p \rrbracket (e) = uu.$$

On montre la réciproque par l'absurde. Soit U une partie infinie de $\mathbb{N}$ qu'on suppose calculatoirement compacte : il y a un programme $\forall_U$.

Soit $\mathbf{i}_{\{0\}} \stackrel{\text{def}}{=} \text{fun } e \rightarrow \text{if } e = 0 \text{ then } uu \text{ else } \text{DIV}_{\text{unit}}$ un programme indicateur de $\{0\}$. Soit $p \stackrel{\text{def}}{=} \text{fun } f \rightarrow \text{fun } e \rightarrow \mathbf{i}_{\{0\}} \ e \ \parallel_{\text{u}} \ f \ (e - 1)$ de type $(\text{nat} \Rightarrow \text{unit}) \Rightarrow (\text{nat} \Rightarrow \text{unit})$. Soit d'une part $q_{\infty} \stackrel{\text{def}}{=} \text{rec } p : \text{nat} \Rightarrow \text{unit}$ et d'autre part, pour chaque $k \in \mathbb{N}$, $q_k \stackrel{\text{def}}{=} p \ (p \ \dots \ (\perp_{\text{nat} \Rightarrow \text{unit}}) \ \dots)$ avec k applications de p . Par construction, la suite des $\llbracket q_k \rrbracket$ est croissante et par définition, $\llbracket q_{\infty} \rrbracket = \sup_{k \in \mathbb{N}} \llbracket q_k \rrbracket$. On montre par récurrence que pour chaque $k \in \mathbb{N}$, q_k est un programme indicateur de $\{0, \dots, k - 1\}$; on en déduit aussi que q_{∞} est un programme indicateur de $\mathbb{N}$.

Soit $k \in \mathbb{N}$. Comme U est une partie infinie de $\mathbb{N}$, U n'est pas majorée, donc il existe $u \in U$ tel que $u \geq k$. Ainsi, $\llbracket q_k \rrbracket (u) = \perp_{\text{unit}}$. Par définition, on a donc $\llbracket \forall_U q_k \rrbracket = \perp_{\text{unit}}$, c'est-à-dire $\llbracket \forall_U \rrbracket (\llbracket q_k \rrbracket) = \perp_{\text{unit}}$. Par continuité, on a donc $\llbracket \forall_U \rrbracket (\llbracket q_{\infty} \rrbracket) = \perp_{\text{unit}}$.

Mais d'autre part, pour tout $u \in U$, $\llbracket q_{\infty} \rrbracket (u) = uu$ car $U \subseteq \mathbb{N}$, donc $\llbracket \forall_U q_{\infty} \rrbracket = uu$, c'est-à-dire $\llbracket \forall_U \rrbracket (\llbracket q_{\infty} \rrbracket) = uu$.

Contradiction.

4 Partie IV

Question 4.1. La suite $(f_{s < k})_{k \in \mathbb{N}}$ est croissante et $f_s = \sup_{k \in \mathbb{N}} f_{s < k}$, donc par continuité, $uu = \llbracket p \rrbracket (f_s) = \sup_{k \in \mathbb{N}} \llbracket p \rrbracket (f_{s < k})$, donc la suite $(\llbracket p \rrbracket (f_{s < k}))_{k \in \mathbb{N}}$ ne peut stationner à $\perp_{\text{unit}}$. D'où le résultat.

Question 4.2. Il est tentant d'écrire que par définition, $\llbracket p/b \rrbracket (f_{s < k}) = \llbracket p \rrbracket (\llbracket b :: f_{s < k} \rrbracket)$, mais ça n'a pas de sens parce que $f_{s < k}$ n'est pas un programme. Remarquons que pour tout $s \in \mathbb{B}^{\mathbb{N}}$ et tout $k \in \mathbb{N}$, la fonction $f_{s < k}$ est calculée par le programme

$$q_{s < k} \stackrel{\text{def}}{=} \text{fun } i \rightarrow \text{if } i = 0 \text{ then } s_0 \text{ else if } i = 1 \text{ then } s_1 \text{ else } \dots \text{ if } i = k - 1 \text{ then } s_{k-1} \text{ else } \perp_{\text{bool}}$$

où i est une variable de type nat et $s_0, \dots, s_{k-1}$ et $k - 1$ sont les constantes syntaxiques adaptées à s et k .

On a donc, pour tout $k \in \mathbb{N}$, $\llbracket p/b \rrbracket (f_{s < k}) = \llbracket p/b \rrbracket q_{s < k} = \llbracket p \rrbracket (\llbracket b :: q_{s < k} \rrbracket) = \llbracket p \rrbracket (\llbracket b :: f_{s < k} \rrbracket)$. Or $\llbracket b :: q_{s < k} \rrbracket$ vaut bien sûr $\llbracket q_{b :: s < 1+k} \rrbracket = f_{b :: s < 1+k}$, d'où le résultat.

Question 4.3. a. On note $\pi_0 = p$ et pour tout $\ell < i$, $\pi_{\ell+1} = \pi_{\ell}/s_{\ell}$. On veut montrer que $K(\pi_i) = \{0\}$. Soit $s' \in \mathbb{B}^{\mathbb{N}}$ et $k \in \mathbb{N}$. On a, pour tout $\ell < i$, $\llbracket \pi_{\ell+1} \rrbracket (f_{s' < k}) = \llbracket \pi_{\ell} \rrbracket (f_{s_{\ell} :: s' < k+1})$. Par récurrence, on obtient que $\llbracket \pi_{\ell+1} \rrbracket (f_{s' < k}) = \llbracket p \rrbracket (f_{s_0 :: \dots :: s_{\ell} :: s' < \ell+1+k})$ et en particulier

$$\llbracket \pi_i \rrbracket (f_{s' < k}) = \llbracket p \rrbracket (f_{s_0 :: \dots :: s_{i-1} :: s' < i+k}),$$

puis

$$\llbracket \pi_i \rrbracket (f_{s' < 0}) = \llbracket p \rrbracket (f_{s_0 :: \dots :: s_{i-1} :: s' < i}).$$

Or, en revenant à la définition, $f_{s_0 :: \dots :: s_{i-1} :: s' < i} = f_{s < i}$, donc $\llbracket \pi_i \rrbracket (f_{s' < 0}) = uu$. Ainsi, $k(s', \pi_i) \leq 0$ puis lui est égal. D'où le résultat.

- b. Supposons $K(p)$ non borné. Soit $A \in \mathbb{N}$: il existe $s \in \mathbb{B}^{\mathbb{N}}$ tel que $k(s, p) \geq A + 1$. Soit $b = s_0$ et s' la suite telle que $s = s_0 :: s'$. Pour tout $k \in \mathbb{N}$, $\llbracket p/b \rrbracket (f_{s' < k}) = \llbracket p \rrbracket (f_{s < k+1})$. Ainsi, $k(s', p/b) = k(s, p) - 1$, donc $k(s', p/b) \geq A$. D'où le résultat.
- c. Par l'absurde, supposons $K(p)$ non borné. En itérant le procédé ci-dessus, on construit une suite $s \in \mathbb{B}^{\mathbb{N}}$ et une suite de programmes π telles que $\pi_0 = p$, $\pi_{\ell+1} = \pi_{\ell}/s_{\ell}$ et $K(\pi_{\ell})$ est non borné. On a $\llbracket p \rrbracket (f_s) = \text{uu}$ par définition de p . Soit donc $i = k(s, p)$. On a vu en (a) que $K(\pi_i) = \{0\}$, qui est borné. Contradiction.

Question 4.4. a. Soit $i = k(b :: s, p)$. On a $i - 1 \in \mathbb{N}$ et $\llbracket p/b \rrbracket (f_{s < i-1}) = \llbracket p \rrbracket (f_{b :: s < i}) = \text{uu}$, donc, par minimalité, $k(s, p/b) \leq i - 1$, d'où le résultat.

- b. Soit $b \in \mathbb{B}$. Remarquons que $K(p/b)$ est une partie bornée non vide de $\mathbb{N}$, donc sa borne supérieure est atteinte. Ainsi, il existe $s \in \mathbb{B}^{\mathbb{N}}$ tel que $k(s, p/b) = |p/b|$. Si $k(b :: s, p) > 0$, alors par la question précédente, $k(s, p/b) < k(b :: s, p)$, donc $|p/b| < k(b :: s, p) \leq |p|$.

Reste à prouver qu'on n'a pas $k(b :: s, p) = 0$. Cela voudrait dire que $\llbracket p \rrbracket (f_{b :: s < 0}) = \text{uu}$. Mais $f_{b :: s < 0}$ est la fonction constante égale à $\perp_{\text{bool}}$; elle ne dépend pas réellement de b et s : par conséquent on aurait $K(p) = \{0\}$ et donc $|p| = 0$ contrairement à l'hypothèse.

Question 4.5. Soit

$$\forall_{\mathbb{C}} \stackrel{\text{def}}{=} \text{rec fun } f \rightarrow \text{fun } p \rightarrow p \text{ (fun } x \rightarrow \perp_{\text{bool}}) \parallel_{\text{u}} (f (p/0) \&_{\text{u}} f (p/1)).$$

On a $\llbracket \forall_{\mathbb{C}} p \rrbracket = \text{uu}$ si et seulement si $\llbracket p \rrbracket (f_{s < 0}) = \text{uu}$ (pour toute suite s , puisque ça ne dépend pas réellement de s) ou, récursivement, pour tout $b \in \mathbb{B}$ à chaque étape (ce qui permet d'énumérer progressivement toutes les suites s), les $p/b / \dots$ finissent par vérifier ce critère.

Un programme p qui vérifie ce critère est tel que $\llbracket p \rrbracket (f) = \text{uu}$ pour tout $f \in \mathbb{C}$. Les questions précédentes ont montré que ce critère est également nécessaire.

La preuve se formalise par récurrence sur le module d'uniforme continuité.