Corrigé de l'examen du cours "Typage et programmation"

D.E.A. "Programmation"

20 décembre 2001

Exercice 1

Question 1.

$$\frac{a \xrightarrow{v} v}{\mathit{pack}(\mathtt{a}) \xrightarrow{\mathtt{v}} \mathtt{pack(v)}} \qquad \frac{a_1 \xrightarrow{v} \mathtt{pack}(v_1) \qquad a_2[x \leftarrow v_1] \xrightarrow{v} v_2}{\mathit{open}\ \mathtt{a_1\ as\ x\ in\ a_2} \xrightarrow{\mathtt{v}} \mathtt{v_2}}$$

Question 2. Supposons $E \vdash \mathtt{open}\ (\mathtt{pack}\ (v))\ \mathtt{as}\ x\ \mathtt{in}\ a : \tau$. D'après les règles de typage (pack) et (open), la dérivation de ce typage est de la forme

$$\frac{\dfrac{E \vdash v : \tau'[\alpha \leftarrow \tau'']}{E \vdash \mathtt{pack}(v) : \exists \alpha.\tau'} \qquad E + \{x : \tau'\} \vdash a : \tau \qquad \alpha \notin \mathcal{L}(\tau) \qquad \alpha \notin \mathcal{L}(E)}{E \vdash \mathtt{open}\ (\mathtt{pack}\ (v))\ \mathtt{as}\ x\ \mathtt{in}\ a : \tau}$$

Considérons la substitution $\varphi = [\alpha \leftarrow \tau'']$. D'après la propriété de stabilité du typage par substitution (proposition 1.2), $E + \{x : \tau'\} \vdash a : \tau$ entraîne $\varphi(E + \{x : \tau'\}) \vdash a : \varphi(\tau)$, c'est-à-dire $\varphi(E) + \{x : \varphi(\tau')\} \vdash a : \varphi(\tau)$. Or, α n'est pas libre dans E ni dans τ. Donc, $\varphi(E) = E$ et $\varphi(\tau) = \tau$. Par conséquent, $E + \{x : \varphi(\tau')\} \vdash a : \tau$.

D'autre part, on sait que $E \vdash v : \varphi(\tau')$. Appliquant le lemme de substitution de valeurs (proposition 2.2) à la substitution de x par v dans a, il vient $E \vdash a[x \leftarrow v] : \tau$. C'est le résultat attendu, car $a[x \leftarrow v]$ est le réduit de $\mathtt{open}\ (\mathtt{pack}\ (v))\ \mathtt{as}\ x\ \mathtt{in}\ a$ par la règle β_{open}.

Question 3. On discute par cas sur la valeur v. Si v est une constante, son type τ est un type de base (`int`, etc). Si v est un opérateur ou une fonction $\mathtt{fun}\ x \to a$, son type est un type fonctionnel $\tau_1 \to \tau_2$. Si v est une paire de valeurs, son type est un type produit $\tau_1 \times \tau_2$. Donc, si v a un type existentiel $\exists \alpha.\tau$ la seule possibilité est que $v = \mathtt{pack}(v')$, et de plus il existe τ' tel que $\emptyset \vdash v' : \tau[\alpha \leftarrow \tau']$.

Question 4. On procède par récurrence sur la structure de a et par cas sur a. Les seuls cas "nouveaux" sont $a = \mathtt{pack}(a')$ ou $a = \mathtt{open}\ a_1\ \mathtt{as}\ x\ \mathtt{in}\ a_2$.

Si $a = \mathtt{pack}(a')$, appliquant l'hypothèse de récurrence à a', il vient que ou bien a' est une valeur v', ou bien a' se réduit. Dans le premier cas, $a = \mathtt{pack}(v')$ est une valeur. Dans le second cas, $a = \mathtt{pack}(a')$ se réduit par la règle de passage au contexte, à l'aide d'un contexte de la forme $\mathtt{pack}(\Gamma)$.

Si $a = \texttt{open}\ a_1\ \texttt{as}\ x\ \texttt{in}\ a_2$, on applique de même l'hypothèse de récurrence à a_1. Si a_1 se réduit, alors a se réduit également par passage à un contexte de la forme $\texttt{open}\ \Gamma\ \texttt{as}\ x\ \texttt{in}\ a_2$. Si a_1 est une valeur v_1, d'après la règle de typage (open), on a $\emptyset \vdash v_1 : \exists\alpha.\tau'$. La question 3 montre que v_1 est nécessairement de la forme $\texttt{pack}(v)$. Donc, a se réduit en tête vers $a_2[x \leftarrow v]$ par la règle β_{open}. Dans les deux cas, a se réduit.

Question 5. Si on enlève la restriction $\alpha \notin \mathcal{L}(\tau)$ dans la règle (open), la variable α peut apparaître libre dans le type du résultat de `open`. Si le `open` apparaît dans la partie gauche d'un `let`, cette variable α pourrait être généralisable, et donc quantifiée universellement dans la partie droite du `let`. Cela permettrait d'instancier α par n'importe quel type dans la partie droite du `let`, ce qui ne respecte pas l'idée initiale que ce α représente un type inconnu, sur lequel on ne sait rien.

Mettons ces considérations en pratique. Prenons un terme simple de type existentiel : $\texttt{pack}(1)$, qui a le type $\exists\alpha.\alpha$. Faisons un `open` dessus dans la partie gauche d'un `let` :

$$\texttt{let}\ y = (\texttt{open}\ (\texttt{pack}(1))\ \texttt{as}\ x\ \texttt{in}\ x)\ \texttt{in}\ \ldots$$

Sans la restriction $\alpha \notin \mathcal{L}(\tau)$, la règle (open) nous permettrait de typer

$$\emptyset \vdash \texttt{open}\ (\texttt{pack}(1))\ \texttt{as}\ x\ \texttt{in}\ x : \alpha$$

et donc on aurait $y : \forall\alpha.\alpha$ dans la partie droite du `let`, alors même que y vaut 1 à l'exécution. Il est alors facile de casser la sûreté du typage, en faisant par exemple

$$\texttt{let}\ y = (\texttt{open}\ (\texttt{pack}(1))\ \texttt{as}\ x\ \texttt{in}\ x)\ \texttt{in}\ y(2)$$

Question 6. Si on enlève la restriction $\alpha \notin \mathcal{L}(E)$ dans la règle (open), on risque de confondre une variable α représentant un type inconnu dans le type d'une variable libre, et la variable α introduite par le `open` représentant le type inconnu dans un type existentiel. Cela permettrait d'utiliser la valeur contenue dans un `pack` de manière non polymorphe par-rapport au type existentiellement quantifié. Par exemple, on pourrait typer

$$\texttt{fun}\ f \to (\texttt{open}\ (\texttt{pack(true)})\ \texttt{as}\ x\ \texttt{in}\ f(x))$$

avec le type suivant :

$$(\alpha \to \alpha) \to \alpha$$

Il suffit en effet de prendre $f : \alpha \to \alpha$ et $\texttt{pack(true)} : \exists\alpha.\alpha$, avec la même variable α. Or, ce type est clairement faux, car la fonction ci-dessus se comporte comme $\texttt{fun}\ f \to f(\texttt{true})$. Une étape de généralisation puis d'instantiation suffit à casser la sûreté du typage :

$$\texttt{let}\ g = \texttt{fun}\ f \to (\texttt{open}\ (\texttt{pack(true)})\ \texttt{as}\ x\ \texttt{in}\ f(x)\ \texttt{in}\ g(\texttt{fun}\ y \to y + 1)$$

car ce terme, bien typé, se réduit en $\texttt{true} + 1$.

Exercice 2

Question 1 Voici une dérivation de typage et de traduction possible :

$$\dfrac{\emptyset \vdash \texttt{cos} : \texttt{float} \to \texttt{float} \Longrightarrow \texttt{cos} \qquad \dfrac{\emptyset \vdash 1 : \texttt{int} \Longrightarrow 1 \qquad \texttt{int} <: \texttt{float}}{\emptyset \vdash 1 : \texttt{float} \Longrightarrow \texttt{floatofint}\ 1}}{\emptyset \vdash \texttt{cos}\ 1 : \texttt{float} \Longrightarrow \texttt{cos}(\texttt{floatofint}\ 1)}$$

La traduction est $\texttt{cos}(\texttt{floatofint}\ 1)$.

Question 2 La fonction $\mathcal{C}(\sigma \to \tau, \sigma' \to \tau')$ doit prendre en argument une fonction f quelconque de type $\sigma \to \tau$ et la transformer en une fonction f' de type $\sigma' \to \tau'$. Comme $\sigma \to \tau <: \sigma' \to \tau'$, on a $\sigma' <: \sigma$ et $\tau <: \tau'$. Puisque $\sigma' <: \sigma$, l'argument de f', de type σ', peut être transformé en valeur de type σ par application de la conversion $\mathcal{C}(\sigma', \sigma)$. Le résultat peut alors être passé en argument à f, qui renvoie un résultat de type τ. Comme $\tau <: \tau'$, ce dernier résultat peut enfin être renvoyé comme résultat de f' avec le type τ' pourvu qu'on lui applique la conversion $\mathcal{C}(\tau, \tau')$. D'où le diagramme suivant :

$$\begin{tikzcd} \sigma \arrow[r, "f"] & \tau \arrow[d, "\mathcal{C}(\tau,\tau')"] \\ \sigma' \arrow[u, "\mathcal{C}(\sigma',\sigma)"] \arrow[r, "f'"'] & \tau' \end{tikzcd}$$

C'est-à-dire :

$$f' = \texttt{fun}\ x \to \mathcal{C}(\tau, \tau')(f(\mathcal{C}(\sigma', \sigma)\ x))$$

Dans le cas d'une conversion de $\texttt{float} \to \texttt{int}$ vers $\texttt{int} \to \texttt{float}$, on obtient :

$$f' = \texttt{fun}\ x \to \texttt{floatofint}(f(\texttt{floatofint}\ x))$$

Question 3 On montre $E \vdash \mathcal{C}(\tau, \tau') : \tau \to \tau'$ pour tout E par récurrence structurelle sur τ et τ'.

Cas $\tau = \texttt{int}$ et $\tau' = \texttt{int}$: $\mathcal{C}(\texttt{int}, \texttt{int})$ est la fonction identité, qui a bien le type $\texttt{int} \to \texttt{int}$.

Cas $\tau = \texttt{float}$ et $\tau' = \texttt{float}$: $\mathcal{C}(\texttt{float}, \texttt{float})$ est la fonction identité, qui a aussi le type $\texttt{float} \to \texttt{float}$.

Cas $\tau = \texttt{int}$ et $\tau' = \texttt{float}$: la conversion $\mathcal{C}(\texttt{int}, \texttt{float})$ est l'opérateur $\texttt{floatofint}$, qui a naturellement le type $\texttt{int} \to \texttt{float}$.

Cas $\tau = \varphi \to \sigma$ et $\tau' = \varphi' \to \sigma'$: on a $\varphi' <: \varphi$ et $\sigma <: \sigma'$, d'où par application de l'hypothèse de récurrence, $E \vdash \mathcal{C}(\varphi', \varphi) : \varphi' \to \varphi$ et $E \vdash \mathcal{C}(\sigma, \sigma') : \sigma \to \sigma'$. Notant $E' = E + \{f : \varphi \to \sigma\} + \{x : \varphi'\}$, on a donc :

$$
\dfrac{\dfrac{\dfrac{E' \vdash \mathcal{C}(\sigma,\sigma') : \sigma \to \sigma' \qquad \dfrac{E' \vdash f : \varphi \to \sigma \qquad \dfrac{E' \vdash \mathcal{C}(\varphi',\varphi) : \varphi' \to \varphi \qquad E' \vdash x : \varphi'}{E' \vdash \mathcal{C}(\varphi',\varphi)\ x : \varphi}}{E' \vdash f(\mathcal{C}(\varphi',\varphi)\ x) : \sigma}}{E' \vdash \mathcal{C}(\sigma,\sigma')(f(\mathcal{C}(\varphi',\varphi)\ x)) : \sigma'}}{E + \{f : \varphi \to \sigma\} \vdash \mathtt{fun}\ x \to \mathcal{C}(\sigma,\sigma')(f(\mathcal{C}(\varphi',\varphi)\ x)) : \varphi' \to \sigma'}}{E \vdash \mathtt{fun}\ f \to \mathtt{fun}\ x \to \mathcal{C}(\tau,\tau')(f(\mathcal{C}(\sigma',\sigma)\ x)) : (\varphi \to \sigma) \to (\varphi' \to \sigma')}
$$

D'où $E \vdash \mathcal{C}(\tau,\tau') : \tau \to \tau'$ comme annoncé.

Question 4. On montre que $E \vdash a : \tau \Longrightarrow b$ implique $E \vdash b : \tau$ par récurrence sur la dérivation de $E \vdash a : \tau \Longrightarrow b$. Si la dernière règle utilisée est (sub), on a une dérivation de la forme suivante :

$$
\dfrac{E \vdash a : \tau' \Longrightarrow b' \qquad \tau' <: \tau}{E \vdash a : \tau \Longrightarrow \mathcal{C}(\tau',\tau)(b')}
$$

avec $b = \mathcal{C}(\tau',\tau)(b')$. Par hypothèse de récurrence appliquée à la prémisse de gauche, il vient $E \vdash b' : \tau'$. Comme par ailleurs $E \vdash \mathcal{C}(\tau',\tau) : \tau' \to \tau$, il vient $E \vdash \mathcal{C}(\tau',\tau)(b') : \tau$ comme attendu.

Si la dérivation se termine par une autre règle de typage-traduction que (sub), le résultat s'ensuit immédiatement de l'hypothèse de récurrence.

Question 5. Dans la question 1, on a vu `cos` comme une fonction de type $\mathtt{float} \to \mathtt{float}$, auquel cas l'argument 1 doit être converti en flottant avant de le passer à la fonction. Mais on peut aussi voir `cos` comme ayant le type $\mathtt{int} \to \mathtt{float}$, auquel cas il n'est pas nécessaire de convertir 1, mais il faut convertir `cos`. La seconde approche correspond à la dérivation suivante :

$$
\dfrac{\dfrac{\emptyset \vdash \mathtt{cos} : \mathtt{float} \to \mathtt{float} \qquad \mathtt{int} \to \mathtt{float} <: \mathtt{float} \to \mathtt{float}}{\emptyset \vdash \mathtt{cos} : \mathtt{int} \to \mathtt{float} \Longrightarrow (\mathtt{fun}\ f \to \mathtt{fun}\ x \to f(\mathtt{floatofint}\ x))\ \mathtt{cos}} \qquad \emptyset \vdash 1 : \mathtt{int} \Longrightarrow 1}{\emptyset \vdash \mathtt{cos}\ 1 : \mathtt{float} \Longrightarrow (\mathtt{fun}\ f \to \mathtt{fun}\ x \to f(\mathtt{floatofint}\ x))\ \mathtt{cos}\ 1}
$$

On obtient donc deux traductions syntaxiquement différentes : $\mathtt{cos}(\mathtt{floatofint}\ 1)$ et $(\mathtt{fun}\ f \to \mathtt{fun}\ x \to f(\mathtt{floatofint}\ x))\ \mathtt{cos}\ 1$. Cependant, la seconde se réduit en la première par deux étapes de β-réduction, et donc les deux traductions calculent le même résultat $\mathtt{cos}(1.0)$.

Question 6 La dérivation initiale ((sub) suivie de (fun)) est de la forme suivante :

$$
\dfrac{\dfrac{E + \{x : \tau'\} \vdash a : \sigma \Longrightarrow b \qquad \sigma <: \tau}{E + \{x : \tau'\} \vdash a : \tau \Longrightarrow \mathcal{C}(\sigma,\tau)\ b}}{E \vdash \mathtt{fun}\ x \to a : \tau' \to \tau \Longrightarrow \mathtt{fun}\ x \to \mathcal{C}(\sigma,\tau)\ b}
$$

Puisque $\sigma <: \tau$, on a $\tau' \to \sigma <: \tau' \to \tau$. D'où la dérivation suivante ((fun) suivie de (sub)) :

$$\dfrac{\dfrac{E + \{x : \tau'\} \vdash a : \sigma \Longrightarrow b}{E \vdash \texttt{fun } x \to a : \tau' \to \sigma \Longrightarrow \texttt{fun } x \to b} \qquad \tau' \to \sigma <: \tau' \to \tau}{E \vdash \texttt{fun } x \to a : \tau' \to \tau \Longrightarrow \mathcal{C}(\tau' \to \sigma, \tau' \to \tau)\ (\texttt{fun } x \to b)}$$

On a

$$b' = \mathcal{C}(\tau' \to \sigma, \tau' \to \tau)\ (\texttt{fun } x \to b) = (\texttt{fun } f \to \texttt{fun } x \to \mathcal{C}(\sigma, \tau)(f((\texttt{fun } y \to y)\ x)))\ (\texttt{fun } x \to b)$$

Après quelques β-réductions, on obtient le résultat attendu :

$$b' \approx \texttt{fun } x \to \mathcal{C}(\sigma, \tau)\ b$$

Question 7 La dérivation initiale ((sub) à gauche de (app)) est de la forme suivante :

$$\dfrac{\dfrac{E \vdash a_1 : \sigma' \to \sigma \Longrightarrow b_1 \qquad \sigma' \to \sigma <: \tau' \to \tau}{E \vdash a_1 : \tau' \to \tau \Longrightarrow \mathcal{C}(\sigma' \to \sigma, \tau' \to \tau)\ b_1} \qquad E \vdash a_2 : \tau' \Longrightarrow b_2}{E \vdash a_1\ a_2 : \tau \Longrightarrow \mathcal{C}(\sigma' \to \sigma, \tau' \to \tau)\ b_1\ b_2}$$

Puisque $\sigma' \to \sigma <: \tau' \to \tau$, on a $\tau' <: \sigma'$ et $\sigma <: \tau$. D'où la dérivation suivante :

$$\dfrac{\dfrac{E \vdash a_1 : \sigma' \to \sigma \Longrightarrow b_1 \qquad \dfrac{E \vdash a_2 : \tau' \Longrightarrow b_2 \qquad \tau' <: \sigma'}{E \vdash a_2 : \sigma' \Longrightarrow \mathcal{C}(\tau', \sigma')\ b_2}}{E \vdash a_1\ a_2 : \sigma \Longrightarrow b_1(\mathcal{C}(\tau', \sigma')\ b_2)} \qquad \sigma <: \tau}{E \vdash a_1\ a_2 : \tau \Longrightarrow \mathcal{C}(\sigma, \tau)(b_1(\mathcal{C}(\tau', \sigma')\ b_2))}$$

On a donc $b' = \mathcal{C}(\sigma, \tau)(b_1(\mathcal{C}(\tau', \sigma')\ b_2))$. D'autre part,

$$\begin{array}{rcl} \mathcal{C}(\sigma' \to \sigma, \tau' \to \tau)\ b_1\ b_2 & = & (\texttt{fun } f \to \texttt{fun } x \to \mathcal{C}(\sigma, \tau)(f(\mathcal{C}(\tau', \sigma')\ x)))\ b_1\ b_2 \\ & \approx & \mathcal{C}(\sigma, \tau)(b_1(\mathcal{C}(\tau', \sigma')\ b_2)) \\ & = & b' \end{array}$$

D'où le résultat annoncé.

Question 8 Par examen des règles de sous-typage, on voit que les seuls moyens d'enchaîner deux sous-typages sont :

$$\begin{array}{l} \texttt{int} <: \texttt{int} <: \texttt{int} \\ \texttt{int} <: \texttt{int} <: \texttt{float} \\ \texttt{int} <: \texttt{float} <: \texttt{float} \\ \texttt{float} <: \texttt{float} <: \texttt{float} \\ \ldots \to \ldots <: \ldots \to \ldots <: \ldots \to \ldots \end{array}$$

D'où une preuve par récurrence structurelle sur τ_1, τ_2 et τ_3, avec deux cas de base (deux types égaux) et un cas inductif (trois types flèches).

Cas $\tau_1 = \tau_2$: dans ce cas, $\tau_1 <: \tau_3$ est équivalent à $\tau_2 <: \tau_3$, et la conversion $\mathcal{C}(\tau_1, \tau_2)$ est la fonction identité. On a donc

$$\texttt{fun}\ x \to \mathcal{C}(\tau_2, \tau_3)\ (\mathcal{C}(\tau_1, \tau_2)\ x) \approx \texttt{fun}\ x \to \mathcal{C}(\tau_2, \tau_3)\ x \approx \mathcal{C}(\tau_2, \tau_3)$$

par β-équivalence et η-équivalence, d'où le résultat annoncé.

Cas $\tau_2 = \tau_3$: même preuve que dans le cas précédent.

Cas $\tau_1 = \varphi_1 \to \sigma_1$ et $\tau_2 = \varphi_2 \to \sigma_2$ et $\tau_3 = \varphi_3 \to \sigma_3$: on a

$$\varphi_3 <: \varphi_2 <: \varphi_1 \quad \text{et} \quad \sigma_1 <: \sigma_2 <: \sigma_3$$

On note $f \circ g$ pour $\texttt{fun}\ x \to f(g\ x)$. Par hypothèse de récurrence, il vient

$$\begin{array}{ll} \varphi_3 <: \varphi_1 & \quad \mathcal{C}(\varphi_3, \varphi_1) \approx \mathcal{C}(\varphi_2, \varphi_1) \circ \mathcal{C}(\varphi_3, \varphi_2) \\ \sigma_1 <: \sigma_3 & \quad \mathcal{C}(\sigma_1, \sigma_3) \approx \mathcal{C}(\sigma_2, \sigma_3) \circ \mathcal{C}(\sigma_1, \sigma_2). \end{array}$$

D'où $\tau_1 <: \tau_3$ par la règle de sous-typage entre types de fonctions. De plus,

$$\begin{array}{rcl} \mathcal{C}(\tau_1, \tau_3) & = & \texttt{fun}\ f \to \mathcal{C}(\sigma_1, \sigma_3) \circ f \circ \mathcal{C}(\varphi_3, \varphi_1) \\ & \approx & \texttt{fun}\ f \to (\mathcal{C}(\sigma_2, \sigma_3) \circ \mathcal{C}(\sigma_1, \sigma_2)) \circ f \circ (\mathcal{C}(\varphi_2, \varphi_1) \circ \mathcal{C}(\varphi_3, \varphi_2)) \\ & \approx & \texttt{fun}\ f \to \mathcal{C}(\sigma_2, \sigma_3) \circ (\mathcal{C}(\sigma_1, \sigma_2)) \circ f \circ \mathcal{C}(\varphi_2, \varphi_1)) \circ \mathcal{C}(\varphi_3, \varphi_2)) \\ & \approx & \texttt{fun}\ f \to \mathcal{C}(\sigma_2, \sigma_3) \circ (\mathcal{C}(\tau_1, \tau_2)\ f) \circ \mathcal{C}(\varphi_3, \varphi_2) \\ & \approx & \texttt{fun}\ f \to \mathcal{C}(\tau_2, \tau_3)(\mathcal{C}(\tau_1, \tau_2)\ f) \end{array}$$

C'est le résultat désiré.

Clarification. Il y avait une erreur dans le sujet : les dérivations normalisées, telles que définies dans le sujet, ne sont pas entièrement déterminées par E et a. On peut en effet avoir deux dérivations des formes suivantes pour une application :

$$\dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée}\\ \vdots\end{array}}{E \vdash a_1 : \tau' \to \tau} \qquad \dfrac{\begin{array}{c}\text{dérivation normalisée}\\ \vdots\end{array}}{E \vdash a_2 : \tau'}}{E \vdash a_1(a_2) : \tau}\ (\text{app})$$

et

$$\dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée}\\ \vdots\end{array}}{E \vdash a_1 : \tau' \to \tau} \qquad \dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée}\\ \vdots\end{array}}{E \vdash a_2 : \tau'} \qquad \tau' <: \tau'}{E \vdash a_2 : \tau'}\ (\text{sub})}{E \vdash a_1(a_2) : \tau}\ (\text{app})$$

Il faut en fait définir les dérivations normalisées par (1) la règle (sub) n'apparaît que juste au dessus à droite de la règle (app), et (2) au dessus à droite d'une règle (app) apparaît forcément une application de la règle (sub). (Comme dans la seconde dérivation ci-dessus mais pas dans la première.)

Question 9. On procède par récurrence sur la dérivation (a priori non normalisée) de $E \vdash a : \tau \Longrightarrow b$, et par cas sur la dernière règle utilisée dans la dérivation.

Cas Axiomes (var), (const) ou (op) : la dérivation est déjà normalisée, il suffit donc de prendre $b' = b$ et $\tau' = \tau$.

Cas Règle (fun) : la dérivation initiale est de la forme suivante :

$$\dfrac{\begin{array}{c}\vdots\\ E_1 \vdash a_1 : \tau_1 \Longrightarrow b_1\end{array}}{E \vdash a : \tau \Longrightarrow b}\ (\text{fun})$$

Appliquant l'hypothèse de récurrence à la prémisse, on obtient une dérivation de la forme suivante :

$$\dfrac{\begin{array}{c}\text{dérivation normalisée } D\\ \vdots\end{array} \qquad \cdots}{E_1 \vdash a_1 : \tau_1 \Longrightarrow b_2}\ (\text{sub})$$

avec $b_2 \approx b_1$. En réappliquant la règle (fun) au résultat de cette dérivation, on obtient :

$$\dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée } D\\ \vdots\end{array} \qquad \cdots}{E_1 \vdash a_1 : \tau_1 \Longrightarrow b_2}\ (\text{sub})}{E \vdash a : \tau \Longrightarrow b_3}\ (\text{fun})$$

avec $b_3 = \texttt{fun}\ x \to b_2 \approx \texttt{fun}\ x \to b_1 = b$. En permutant les règles (sub) et (fun) comme à la question 6, il vient une dérivation de la forme suivante :

$$\dfrac{\dfrac{\begin{array}{c}\text{(dérivation normalisée } D)\\ \vdots\end{array}}{\cdots}\ (\text{fun}) \qquad \cdots}{E \vdash a : \tau \Longrightarrow b_4}\ (\text{sub})$$

avec $b_4 \approx b_3$. La sous-dérivation composée de D et de la règle (fun) étant normalisée, la dérivation ci-dessus convient, avec $b' = b_4 \approx b$ par transitivité de $\approx$.

Cas Règle (app) : appliquant l'hypothèse de récurrence aux deux prémisses de la règle (app), et en recombinant avec (app) les dérivations obtenues, il vient une dérivation de la forme suivante :

$$\dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée } D_1\\ \vdots \qquad \cdots\end{array}}{\cdots}\ (\text{sub}) \qquad \dfrac{\begin{array}{c}\text{dérivation normalisée } D_2\\ \vdots \qquad \cdots\end{array}}{\cdots}\ (\text{sub})}{E \vdash a : \tau \Longrightarrow b_1}\ (\text{app})$$

avec $b_1 \approx b$. En permutant (app) et (sub) à gauche comme à la question 7, il vient :

$$\dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée } D_1\\ \vdots\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée } D_2\\ \vdots \qquad \cdots\end{array}}{\cdots}\ (\text{sub})}{\cdots}\ (\text{sub})}{\cdots \qquad \cdots}\ (\text{app})}{E \vdash a : \tau \Longrightarrow b_2}\ (\text{sub})$$

avec $b_2 \approx b_1$. Par transitivité du sous-typage (question 8), on peut combiner les deux étapes de sous-typage à droite en une seule :

$$\dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée } D_1\\ \vdots\end{array} \qquad \dfrac{\begin{array}{c}\text{dérivation normalisée } D_2\\ \vdots \qquad \cdots\end{array}}{\cdots}\ (\text{sub})}{\cdots \qquad \cdots}\ (\text{app})}{E \vdash a : \tau \Longrightarrow b_3}\ (\text{sub})$$

avec $b_3 \approx b_2$. La dérivation ci-dessus est bien de la forme désirée : une dérivation normalisée suivie d'une étape (sub).

Cas Règle (sub) : on applique l'hypothèse de récurrence à la prémisse. Il vient une dérivation de la forme suivante :

$$\dfrac{\dfrac{\begin{array}{c}\text{dérivation normalisée } D\\ \vdots \qquad \cdots\end{array}}{\cdots \qquad \cdots}\ (\text{sub})}{E \vdash a : \tau \Longrightarrow b_1}\ (\text{sub})$$

avec $b_1 \approx b$. On combine alors les deux étapes (sub) en une seule à l'aide de la question 8, et on obtient une dérivation de la forme attendue.

Question 10. En appliquant la question 9 aux deux dérivations de $E \vdash a : \tau \Longrightarrow b_1$ et $E \vdash a : \tau \Longrightarrow b_2$, il vient des dérivations de la forme suivante :

$$\dfrac{\dfrac{\text{dérivation normalisée } D_1 \atop \vdots}{E \vdash a : \tau_1 \Longrightarrow b'_1} \qquad \tau_1 <: \tau}{E \vdash a : \tau \Longrightarrow \mathcal{C}(\tau_1, \tau)\ b'_1} \qquad\qquad \dfrac{\dfrac{\text{dérivation normalisée } D_2 \atop \vdots}{E \vdash a : \tau_1 \Longrightarrow b'_2} \qquad \tau_2 <: \tau}{E \vdash a : \tau \Longrightarrow \mathcal{C}(\tau_2, \tau)\ b'_2}$$

Comme les dérivations normalisées sont entièrement déterminées par E et a, les dérivations D_1 et D_2 sont identiques, et donc $\tau_1 = \tau_2$ et $b'_1 = b'_2$. Comme $b_1 \approx \mathcal{C}(\tau_1, \tau)\ b'_1$ et $b_2 \approx \mathcal{C}(\tau_2, \tau)\ b'_2$, il s'ensuit $b_1 \approx b_2$.