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

D.E.A. "Programmation"

9 décembre 1999

Exercice

Question 1. Cas des règles (1) et (2) : on a $a = \texttt{hastype}_{\tau'}(\texttt{dyn}_{\tau''}(v))$ avec $a' = \texttt{true}$ ou $a' = \texttt{false}$. Vu le type de $\texttt{hastype}_{\tau'}$, on a forcément $\tau = \texttt{bool}$. Or, `true` et `false` sont deux constantes de type `bool`. Donc, $E \vdash a' : \tau$ et (H1) est vérifiée.

Cas de la règle (3) : on a $a = \texttt{coerce}_{\tau'}(\texttt{dyn}_{\tau''}(v))$ avec $\tau' = \tau''$ et $a' = v$. Vu les types de `coerce` et de `dyn`, la dérivation de typage de a est nécessairement de la forme suivante :

$$\dfrac{E \vdash \texttt{coerce}_{\tau'} : \texttt{dyn} \to \tau' \qquad \dfrac{E \vdash \texttt{dyn}_{\tau''} : \tau'' \to \texttt{dyn} \qquad E \vdash v : \tau''}{E \vdash \texttt{dyn}_{\tau''}(v) : \texttt{dyn}}}{E \vdash \texttt{coerce}_{\tau'}(\texttt{dyn}_{\tau''}(v)) : \tau'}$$

D'où $\tau = \tau'$. De plus, on a une dérivation de typage de $E \vdash v : \tau''$. Comme $a' = v$ et $\tau = \tau' = \tau''$, il vient $E \vdash a' : \tau$ comme désiré, et (H1) est vérifiée.

Question 2. Supposons $\emptyset \vdash \texttt{hastype}_{\tau}(v) : \tau'$. Vu le type de $\texttt{hastype}_{\tau}$, la dérivation de ce typage est de la forme :

$$\dfrac{E \vdash \texttt{hastype}_{\tau} : \texttt{dyn} \to \texttt{bool} \qquad E \vdash v : \texttt{dyn}}{E \vdash \texttt{hastype}_{\tau}(v) : \texttt{bool}}$$

Par examen des différentes règles de typage qui peuvent s'appliquer à une valeur, on voit que la valeur v de type `dyn` est nécessairement de la forme $\texttt{dyn}_{\tau'}(v')$. (En effet, les autres valeurs sont ou bien des constantes, de type `int`, `bool`, `string`, etc ; ou bien des fonctions ou des opérateurs, de type $\tau_1 \to \tau_2$; ou bien des paires de valeurs, de type $\tau_1 \times \tau_2$.) Donc, $a = \texttt{hastype}_{\tau}(\texttt{dyn}_{\tau'}(v'))$. Selon que $\tau' = \tau$ ou $\tau' \neq \tau$, l'une des règles (2) ou (3) s'applique et permet de réduire a.

Question 3. Non, l'hypothèse (H2) n'est pas vérifiée pour l'opérateur $\texttt{coerce}_{\tau}$. En effet, considérons $a = \texttt{coerce}_{\texttt{bool}}(\texttt{dyn}_{\texttt{int}}(1))$. Cette expression a est de la forme $\texttt{coerce}_{\tau}$ appliqué à une valeur. Elle est bien typée, puisqu'on a la dérivation suivante :

$$\dfrac{E \vdash \texttt{coerce}_{\texttt{bool}} : \texttt{dyn} \to \texttt{bool} \qquad \dfrac{E \vdash \texttt{dyn}_{\texttt{int}} : \texttt{int} \to \texttt{dyn} \qquad E \vdash 1 : \texttt{int}}{E \vdash \texttt{dyn}_{\texttt{int}}(1) : \texttt{dyn}}}{E \vdash \texttt{coerce}_{\texttt{bool}}(\texttt{dyn}_{\texttt{int}}(1)) : \texttt{bool}}$$

Cependant, aucune règle de réduction ne s'applique à a. En particulier, (3) ne s'applique pas car $\texttt{int} \neq \texttt{bool}$. Donc, (H2) n'est pas vérifiée.

Une approche pour corriger ce problème est d'ajouter une règle de réduction pour des termes de la forme $a = \texttt{coerce}_\tau(\texttt{dyn}_{\tau'}(v))$ dans le cas où $\tau \neq \tau'$. Bien sûr, il ne faut pas ajouter la règle

$$\texttt{coerce}_\tau(\texttt{dyn}_{\tau'}(v)) \xrightarrow{\varepsilon} v$$

car cela casserait la sûreté du typage dans le cas $\tau \neq \tau'$: on aurait par exemple $\texttt{coerce}_{\texttt{bool}}(\texttt{dyn}_{\texttt{int}}(1)) \xrightarrow{\varepsilon} 1$ alors que le terme de gauche a le type `bool`, et donc cette règle permettrait de manipuler 1 avec le type `bool`. (C'est pour cette raison exactement que le “cast” de C, qui ne vérifie pas les types à l'exécution, n'est pas sûr.)

Si l'on dispose d'exceptions, comme au chapitre 5 du cours, la règle suivante fait l'affaire :

$$\texttt{coerce}_\tau(\texttt{dyn}_{\tau'}(v)) \xrightarrow{\varepsilon} \texttt{raise}(v_{\texttt{coerce}})$$

où $v_{\texttt{coerce}}$ est une valeur d'exception particulière. (C'est la sémantique du “cast” de Java, où le type est vérifié dynamiquement et une exception `ClassCastException` est levée si le type est incorrect.) En effet, le membre de droite, $\texttt{raise}(v_{\texttt{coerce}})$, appartient à tous les types, et en particulier au type τ du membre de gauche.

En l'absence d'exceptions, on peut simplement choisir de “boucler” :

$$\texttt{coerce}_\tau(\texttt{dyn}_{\tau'}(v)) \xrightarrow{\varepsilon} \texttt{coerce}_\tau(\texttt{dyn}_{\tau'}(v))$$

Tout programme qui rencontre une erreur de type dynamique en exécutant un `coerce` ne termine jamais. Mais cela n'invalide ni (H1) ni (H2), et donc la sûreté du typage statique est respectée.

Une autre approche est de remplacer `hastype` et `coerce` par une construction unique qui effectue simultanément le test de type et l'extraction de la valeur contenue dans le dynamique, dans le style du filtrage sur les types concrets. Une telle construction est de la forme

`match-dynamic` a `with` $\texttt{dyn}_\tau$`(x) ->` b `| _ ->` c

Si le type contenu dans le dynamique a est égal à τ, la valeur contenue dans le dynamique est liée à x et b est évaluée. Sinon, c est évaluée. Cette construction peut être vue comme du sucre syntaxique pour l'opérateur MD_τ dont le type est

$$MD_\tau : \forall\alpha.\ \texttt{dyn} \times (\tau \to \alpha) \times (\texttt{unit} \to \alpha) \to \alpha$$

et les règles de réduction sont :

$$\begin{array}{lcll} MD_\tau(\texttt{dyn}_{\tau'}(v), v_1, v_2) & \xrightarrow{\varepsilon} & v_1\ v & \text{si } \tau = \tau' \\ MD_\tau(\texttt{dyn}_{\tau'}(v), v_1, v_2) & \xrightarrow{\varepsilon} & v_2\ () & \text{si } \tau \neq \tau' \end{array}$$

Question 4. Avec les schémas de types donnés pour $\texttt{dyn}_{\alpha\to\alpha}$ et $\texttt{coerce}_{\alpha\to\alpha}$, la dérivation de typage suivante est valide :

$$\dfrac{\dfrac{\texttt{bool} \to \texttt{bool} \leq \forall\alpha.\ \texttt{dyn} \to (\alpha \to \alpha)}{E \vdash \texttt{coerce}_{\alpha\to\alpha} : \texttt{dyn} \to (\texttt{bool} \to \texttt{bool})} \quad \dfrac{\dfrac{(\texttt{int} \to \texttt{int}) \to \texttt{dyn} \leq \forall\alpha.\ (\alpha \to \alpha) \to \texttt{dyn}}{E \vdash \texttt{dyn}_{\alpha\to\alpha} : (\texttt{int} \to \texttt{int}) \to \texttt{dyn}} \quad E \vdash \texttt{fun}\ x \to x + 1 : \texttt{int} \to \texttt{int}}{E \vdash \texttt{dyn}_{\alpha\to\alpha}(\texttt{fun}\ x \to x + 1) : \texttt{dyn}}}{E \vdash \texttt{coerce}_{\alpha\to\alpha}(\texttt{dyn}_{\alpha\to\alpha}(\texttt{fun}\ x \to x + 1)) : \texttt{bool} \to \texttt{bool}}$$

Cependant, le terme $\texttt{coerce}_{\alpha\to\alpha}(\texttt{dyn}_{\alpha\to\alpha}(\texttt{fun}\ x \to x + 1))$ s'évalue en $\texttt{fun}\ x \to x + 1$, qui n'est pas une valeur du type $\texttt{bool} \to \texttt{bool}$.

Problème

Question 1. La dérivation de typage suivante est valide :

$$\dfrac{\dfrac{(\alpha \to \alpha) \to \alpha \leq TC(\mathtt{fix})}{\emptyset \vdash \mathtt{fix} : (\alpha \to \alpha) \to \alpha} \qquad \dfrac{\{x : \alpha\} \vdash x : \alpha}{\emptyset \vdash \mathtt{fun}\ x \to x : \alpha \to \alpha}}{\emptyset \vdash \mathtt{fix}(\mathtt{fun}\ x \to x) : \alpha}$$

D'autre part, la règle de réduction $\delta_{\mathtt{fix}}$ nous dit que :

$$\mathtt{fix}(\mathtt{fun}\ x \to x) \to x[x \leftarrow \mathtt{fix}(\mathtt{fun}\ x \to x)] = \mathtt{fix}(\mathtt{fun}\ x \to x)$$

Question 2. Supposons $a \to a_1 \to a_2 \to \ldots \to v$. Par le théorème de préservation du typage par les réductions, on a $a \sqsubseteq a_1 \sqsubseteq a_2 \sqsubseteq \ldots \sqsubseteq v$. Comme $\emptyset \vdash a : \alpha$, cela signifie que $\emptyset \vdash a_1 : \alpha$, puis $\emptyset \vdash a_2 : \alpha$, et enfin $\emptyset \vdash v : \alpha$. Or, il n'existe par de valeur dont le type est α. En effet, une valeur est soit une constante (de type `int`), soit une fonction ou un opérateur (dont le type est forcément un type flèche), soit une paire de valeurs (dont le type est forcément un type produit). D'où contradiction.

Question 3. Un premier exemple est $a = \omega$. (En effet, par la propriété de stabilité du typage par substitution, $\emptyset \vdash \omega : \alpha$ entraîne $\emptyset \vdash \omega : \tau$ pour tout type τ, et en particulier $\tau = \beta \to \alpha$.) Un second exemple est $a = \mathtt{fun}\ x \to \omega$.

Question 4. Soit b une expression close bien typée quelconque et τ son type. Toujours par stabilité du typage par substitution, $\emptyset \vdash a : \beta \to \alpha$ entraîne $\emptyset \vdash a : \tau \to \alpha$. Donc, $\emptyset \vdash a\ b : \alpha$. Par la question 2, il s'ensuit que $a\ b$ diverge forcément.

Question 5. Voici trois exemples d'expressions closes de type $\alpha \to \alpha$:
- ω ;
- $\mathtt{fun}\ x \to \omega$;
- $\mathtt{fun}\ x \to x$.

Question 6. Puisque deux valeurs sont équivalentes en `int` si et seulement si ce sont la même constante entière, deux fonctions f et f' sont équivalentes en $\mathtt{int} \to \mathtt{int}$ si et seulement si, pour tout entier n, ou bien $f\ n$ diverge et $f'\ n$ diverge, ou bien il existe un entier m tel que $f\ n \overset{*}{\to} m$ et $f'\ n \overset{*}{\to} m$.

Les deux fonctions $\mathtt{fun}\ x \to x$ et $\mathtt{fun}\ x \to x + 1$ ne sont donc pas équivalentes en $\mathtt{int} \to \mathtt{int}$. En effet, la première fonction appliquée à l'entier 1 se réduit en 1, alors que la seconde appliquée à l'entier 1 également se réduit en 2.

En revanche, les deux fonctions $\mathtt{fun}\ x \to x + x$ et $\mathtt{fun}\ x \to x * 2$ sont bien équivalentes en $\mathtt{int} \to \mathtt{int}$, car appliquées à un entier n quelconque, elles renvoient toutes deux l'entier $2n$.

Question 7. Pour réduire (a, b), on réduit d'abord a en une valeur, puis b en une valeur.

Si a diverge, on a une séquence de réduction infinie $a \to a_1 \to a_2 \to \ldots$. Par passage au contexte à gauche de la paire, on en déduit la séquence de réductions

$$(a, b) \to (a_1, b) \to (a_2, b) \to \ldots$$

qui est également infinie. Donc, (a, b) diverge.

Si $a \xrightarrow{*} v$ et b diverge, on a une séquence de réduction finie $a \to a_1 \to a_2 \to \ldots \to v$ et une séquence infinie $b \to b_1 \to b_2 \to \ldots$. Par passage au contexte à gauche de la paire puis à droite, il vient la séquence de réductions

$$(a, b) \to (a_1, b) \to (a_2, b) \to \ldots \to (v, b) \to (v, b_1) \to (v, b_2) \to \ldots$$

qui est infinie. Donc, (a, b) diverge également.

Enfin, si $a \xrightarrow{*} v$ et $b \xrightarrow{*} w$, on a des réductions finies $a \to a_1 \to a_2 \to \ldots \to v$ et $b \to b_1 \to b_2 \to \ldots \to w$, qui permettent de construire la séquence suivante pour (a, b) :

$$(a, b) \to (a_1, b) \to (a_2, b) \to \ldots \to (v, b) \to (v, b_1) \to (v, b_2) \to \ldots \to (v, w)$$

Donc, (a, b) s'évalue en (v, w).

Supposons maintenant $\rho \vdash a \approx a' : \tau_1$ et $\rho \vdash b \approx b' : \tau_2$.

Si a diverge, par définition de l'équivalence, a' diverge également. Par les remarques ci-dessus, cela veut dire que (a, b) et (a', b') divergent aussi, et donc sont équivalentes en $\tau_1 \times \tau_2$.

Si a ne diverge pas mais b diverge, par définition de l'équivalence, cela entraîne que a' ne diverge pas mais que b' diverge. Donc, (a, b) et (a', b') divergent toutes deux, et donc sont équivalentes en $\tau_1 \times \tau_2$.

Si $a \xrightarrow{*} v$ et $b \xrightarrow{*} w$, par définition de l'équivalence entre expressions, il existe v' et w' telles que $a' \xrightarrow{*} v'$ et $b' \xrightarrow{*} w'$. Par les considérations ci-dessus, cela veut dire que $a\ b \xrightarrow{*} (v, w)$ et $a'\ b' \xrightarrow{*} (v', w')$. De plus, $\rho \vdash v \approx v' : \tau_1$ et $\rho \vdash w \approx w' : \tau_2$. Par définition de l'équivalence en un type produit, on a donc $\rho \vdash (v, w) \approx (v', w') : \tau_1 \to \tau_2$. Il s'ensuit que $\rho \vdash (a, b) \approx (a', b') : \tau_1 \to \tau_2$.

Question 8. Pour réduire $a\ b$, on réduit d'abord a en une valeur, puis b en une valeur, puis on effectue une étape de β-réduction.

Par le même raisonnement qu'à la question 7, on voit que si a diverge, alors $a\ b$ diverge, et si a converge mais b diverge, $a\ b$ diverge également.

Enfin, si $a \xrightarrow{*} v$ et $b \xrightarrow{*} w$, on a des réductions finies $a \to a_1 \to a_2 \to \ldots \to v$ et $b \to b_1 \to b_2 \to \ldots \to w$, qui permettent de construire la séquence suivante pour $a\ b$:

$$a\ b \to a_1\ b \to a_2\ b \to \ldots \to v\ b \to v\ b_1 \to v\ b_2 \to \ldots \to v\ w$$

La réduction se poursuit alors soit par une réduction β_{fun}, soit par une δ-règle.

Supposons maintenant $\rho \vdash a \approx a' : \tau_1 \to \tau_2$ et $\rho \vdash b \approx b' : \tau_1$.

Si a diverge, par définition de l'équivalence, a' diverge également. Par les remarques ci-dessus, cela veut dire que $a\ b$ et $a'\ b'$ divergent aussi, et donc sont équivalentes en τ_2.

Si a ne diverge pas mais b diverge, par définition de l'équivalence, cela entraîne que a' ne diverge pas mais que b' diverge. Donc, $a\ b$ et $a'\ b'$ divergent toutes deux, et donc sont équivalentes en τ_2.

Si $a \xrightarrow{*} v$ et $b \xrightarrow{*} w$, par définition de l'équivalence entre expressions, il existe v' et w' telles que $a' \xrightarrow{*} v'$ et $b' \xrightarrow{*} w'$. Par les considérations ci-dessus, cela veut dire que $a\ b \xrightarrow{*} v\ w$ et $a'\ b' \xrightarrow{*} v'\ w'$.

De plus, $\rho \vdash v \approx v' : \tau_1 \to \tau_2$ et $\rho \vdash w \approx w' : \tau_1$. Par définition de l'équivalence en un type fonctionnel, on a donc $\rho \vdash v\ w \approx v'\ w' : \tau_2$. De deux choses l'une : soit $v\ w$ et $v'\ w'$ divergent toutes deux, et alors $a\ b$ et $a'\ b'$ divergent également ; soit $v\ w$ et $v'\ w'$ convergent sur deux valeurs équivalentes en τ_2, et alors $a\ b$ et $a'\ b'$ convergent sur ces deux mêmes valeurs équivalentes en τ_2. Dans les deux cas, on a bien $\rho \vdash a\ b \approx a'\ b' : \tau_2$.

Question 9. Écrivons σ sous la forme $\forall \alpha_1 \ldots \alpha_n.\ \tau'$. Puisque τ est instance de σ, il existe des types $\tau_1, \ldots, \tau_n$ tels que $\tau = \tau'[\alpha_1 \leftarrow \tau_1, \ldots, \alpha_n \leftarrow \tau_n]$. Par définition de l'équivalence en un schéma, $\rho \vdash a \approx a' : \sigma$ entraîne $\rho + \{\alpha_1 \leftarrow R_1; \ldots; \alpha_n \leftarrow R_n\} \vdash a \approx a' : \tau'$ pour tous ensembles de paires de valeurs $R_1 \ldots R_n$. En particulier, prenant $R_i = R(\tau_i)$, il vient

$$\rho + \{\alpha_1 \leftarrow R(\tau_1), \ldots, \alpha_n \leftarrow R(\tau_n)\} \vdash a \approx a' : \tau'$$

Et par le lemme de substitution, cela entraîne

$$\rho \vdash a \approx a' : \tau'[\alpha_1 \leftarrow \tau_1, \ldots, \alpha_n \leftarrow \tau_n]$$

c'est-à-dire $\rho \vdash a \approx a' : \tau$.

Question 10. La preuve est par récurrence sur la dérivation de $E \vdash a : \tau$ et par cas sur a. On abrège les substitutions $[x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n]$ et $[x_1 \leftarrow v'_1, \ldots, x_n \leftarrow v'_n]$ par $[\vec{x} \leftarrow \vec{v}]$ et $[\vec{x} \leftarrow \vec{v}']$ respectivement.

Cas a est un identificateur x. Vu la règle de typage (inst), on a $x \in \mathrm{Dom}(E)$ et $\tau \leq E(x)$. Donc, x est l'un des x_i, et $a[\vec{x} \leftarrow \vec{v}] = v_i$ et $a[\vec{x} \leftarrow \vec{v}'] = v'_i$. Il faut donc montrer $\rho \vdash v_i \approx v'_i : \tau$. C'est une conséquence immédiate du résultat de la question 9, car $\rho \vdash v_i \approx v'_i : E(x_i)$ par hypothèse, et τ est instance de $E(x)$.

(Attention, presque tout le monde a supposé $\tau = E(x_i)$; ce n'est pas forcément vrai avec les règles de typage de mini-ML.)

Cas a est une constante entière n. On a donc $\tau = \mathtt{int}$, $a[\vec{x} \leftarrow \vec{v}] = n$ et $a[\vec{x} \leftarrow \vec{v}'] = n$. Le résultat attendu est donc $\rho \vdash n \approx n : \mathtt{int}$, et il est trivialement vrai vu la définition de l'équivalence en le type `int`.

Cas a est un opérateur op. Admis. (Il suffit de montrer $\emptyset \vdash op \approx op : TC(op)$ pour chaque opérateur op, puis d'appliquer la question 9.)

Cas a est une fonction $\mathtt{fun}\ x \to b$. D'après la règle (fun), on a $\tau = \tau_1 \to \tau_2$ et $E + \{x : \tau_1\} \vdash b : \tau_2$. Quitte à renommer x, on peut supposer que x est distincte des x_i. Soient w, w' deux valeurs telles que $\rho \vdash w \approx w' : \tau_1$. On applique l'hypothèse de récurrence à b avec l'environnement $E + \{x : \tau_1\}$ et les deux substitutions $[\vec{x} \leftarrow \vec{v}; x \leftarrow w]$ et $[\vec{x} \leftarrow \vec{v}'; x \leftarrow w']$. Il vient

$$\rho \vdash b[\vec{x} \leftarrow \vec{v}; x \leftarrow w] \approx b[\vec{x} \leftarrow \vec{v}'; x \leftarrow w'] : \tau_2$$

c'est-à-dire

$$\rho \vdash (b[\vec{x} \leftarrow \vec{v}])[x \leftarrow w] \approx (b[\vec{x} \leftarrow \vec{v}'])[x \leftarrow w'] : \tau_2$$

Maintenant, un petit lemme d'usage général :

Si $a \to b$ et $a' \to b'$ et $\rho \vdash b \approx b' : \tau$, alors $\rho \vdash a \approx a' : \tau$

En effet, ou bien b et b' divergent, auquel cas a et a' aussi ; ou bien b et b' convergent, auquel cas a converge sur la même valeur que b et a' sur la même valeur que b'.

Appliquant le lemme aux β-réductions $(\mathtt{fun}\ x \to b[\vec{x} \leftarrow \vec{v}])\ w \to (b[\vec{x} \leftarrow \vec{v}])[x \leftarrow w]$ et $(\mathtt{fun}\ x \to b[\vec{x} \leftarrow \vec{v}'])\ w' \to (b[\vec{x} \leftarrow \vec{v}'])[x \leftarrow w']$, il s'ensuit

$$\rho \vdash (\mathtt{fun}\ x \to b[\vec{x} \leftarrow \vec{v}])\ w \approx (\mathtt{fun}\ x \to b[\vec{x} \leftarrow \vec{v}'])\ w' : \tau_2$$

Ceci étant vrai pour toutes valeurs w, w' telles que $\rho \vdash w \approx w' : \tau_1$, on en déduit par définition de l'équivalence entre valeurs en un type fonctionnel, que

$$\rho \vdash (\texttt{fun}\ x \rightarrow b)[\vec{x} \leftarrow \vec{v}] \approx (\texttt{fun}\ x \rightarrow b)[\vec{x} \leftarrow \vec{v}'] : \tau_1 \rightarrow \tau_2$$

C'est le résultat attendu.

Cas a est une application $b\ c$. D'après la règle (app), on a $E \vdash b : \tau' \rightarrow \tau$ et $E \vdash c : \tau'$. Appliquant l'hypothèse de récurrence à b et à c, il vient

$$\rho \vdash b[\vec{x} \leftarrow \vec{v}] \approx b[\vec{x} \leftarrow \vec{v}'] : \tau' \rightarrow \tau \qquad \rho \vdash c[\vec{x} \leftarrow \vec{v}] \approx c[\vec{x} \leftarrow \vec{v}'] : \tau'$$

Le résultat attendu découle de la question 8.

Cas a est une paire (b, c). Même raisonnement que le cas précédent, en utilisant la question 7 au lieu de la 8.

Cas a est $\texttt{let}\ x = b\ \texttt{in}\ c$.

On commence par montrer un résultat similaire à ceux des questions 7 et 8 pour le cas du `let` : si $\rho \vdash a \approx a' : \sigma$ et si $\rho \vdash b[x \leftarrow v] \approx b'[x \leftarrow v'] : \tau$ pour toutes valeurs v, v' telles que $\rho \vdash v \approx v' : \sigma$, alors $\rho \vdash (\texttt{let}\ x = a\ \texttt{in}\ b) \approx (\texttt{let}\ x = a'\ \texttt{in}\ b') : \tau$.

Si a diverge, on a une séquence de réductions infinies

$$(\texttt{let}\ x = a\ \texttt{in}\ b) \rightarrow (\texttt{let}\ x = a_1\ \texttt{in}\ b) \rightarrow (\texttt{let}\ x = a_2\ \texttt{in}\ b) \rightarrow \ldots$$

et donc $\texttt{let}\ x = a\ \texttt{in}\ b$ diverge. Si $a \overset{*}{\rightarrow} v$, on a

$$(\texttt{let}\ x = a\ \texttt{in}\ b) \overset{*}{\rightarrow} (\texttt{let}\ x = v\ \texttt{in}\ b) \rightarrow b[x \leftarrow v]$$

et donc $(\texttt{let}\ x = a\ \texttt{in}\ b)$ s'évalue comme $b[x \leftarrow v]$.

Supposons maintenant $\rho \vdash a \approx a' : \sigma$ et $\rho \vdash b[x \leftarrow v] \approx b'[x \leftarrow v'] : \tau$ pour toutes valeurs v, v' telles que $\rho \vdash v \approx v' : \sigma$.

Si a diverge, a' diverge également puisqu'elles sont équivalentes en σ, et donc $(\texttt{let}\ x = a\ \texttt{in}\ b)$ et $(\texttt{let}\ x = a'\ \texttt{in}\ b')$ sont équivalentes en τ puisque toutes deux divergentes.

Si $a \overset{*}{\rightarrow} v$, il existe v' telle que $a' \overset{*}{\rightarrow} v'$, et de plus $\rho \vdash v \approx v' : \sigma$. Par hypothèse sur b et b', on a donc $\rho \vdash b[x \leftarrow v] \approx b'[x \leftarrow v'] : \tau$, et puisque $(\texttt{let}\ x = a\ \texttt{in}\ b)$ et $(\texttt{let}\ x = a'\ \texttt{in}\ b')$ s'évalue comme $b[x \leftarrow v]$ et $b'[x \leftarrow v']$ respectivement, il s'ensuit $\rho \vdash (\texttt{let}\ x = a\ \texttt{in}\ b) \approx (\texttt{let}\ x = a'\ \texttt{in}\ b') : \tau$.

Une fois ce résultat montré, revenons au cas `let` de la récurrence. Là aussi, on peut supposer que x est distincte des x_i. Les prémisses de la règle (let) sont vraies :

$$E \vdash b : \tau' \qquad E + \{x : Gen(\tau', E)\} \vdash c : \tau$$

Écrivons $Gen(\tau', E)$ sous la forme $\forall \alpha_1 \ldots \alpha_k.\ \tau'$ avec $\{\alpha_1 \ldots \alpha_k\} = FV(\tau') \setminus FV(E)$. Soient $R_1, \ldots, R_k$ des ensembles arbitraires de paires de valeurs. Appliquant l'hypothèse de récurrence à b et à l'interprétation $\rho' = \rho + \{\alpha_1 \leftarrow R_1; \ldots; \alpha_k \leftarrow R_k\}$, il vient $\rho' \vdash b[\vec{x} \leftarrow \vec{v}] \approx b[\vec{x} \leftarrow \vec{v}'] : \tau'$. Ceci valant pour des $R_1 \ldots R_k$ quelconques, il s'ensuit que $\rho \vdash b[\vec{x} \leftarrow \vec{v}] \approx b[\vec{x} \leftarrow \vec{v}'] : Gen(\tau', E)$. Soient maintenant des valeurs w, w' telles que $\rho \vdash \rho \vdash w \approx w' : Gen(\tau, E)$. Appliquant l'hypothèse de récurrence à c avec l'environnement $E + \{x : Gen(\tau', E)\}$ et les deux substitutions $[\vec{x} \leftarrow \vec{v}; x \leftarrow w]$ et $[\vec{x} \leftarrow \vec{v}'; x \leftarrow w']$, il vient $\rho \vdash c[\vec{x} \leftarrow \vec{v}; x \leftarrow w] \approx c[\vec{x} \leftarrow \vec{v}'; x \leftarrow w'] : \tau$. Le résultat attendu découle alors de la propriété montrée plus haut.

Question 11. Soit A un ensemble quelconque de paires de valeurs. Le lemme fondamental dit que $\{\alpha \leftarrow A\} \vdash a \approx a : \alpha \rightarrow \alpha$. De deux choses l'une : ou bien a diverge, ou bien $a \xrightarrow{*} f$ avec $\{\alpha \leftarrow A\} \vdash f \approx f : \alpha \rightarrow \alpha$. Cela signifie que pour tous $(v, v') \in A$, ou bien $f\ v$ et $f\ v'$ divergent, ou bien $f\ v \xrightarrow{*} w$, $f\ v' \xrightarrow{*} w'$, et $(w, w') \in A$.

Soit maintenant v une valeur quelconque. Prenons $A = \{(0, v)\}$. Puisque par construction $(0, v) \in A$, nous obtenons que ou bien $f\ 0$ et $f\ v$ divergent, ou bien $f\ 0 \xrightarrow{*} w_0$ et $f\ v \xrightarrow{*} w$ et $(w_0, w) \in A$, c'est-à-dire $w_0 = 0$ et $w = v$ puisque A contient un seul élément.

Donc, si $f\ 0$ diverge, alors $f\ v$ diverge pour tout v ; et si $f\ 0$ ne diverge pas, alors $f\ v \xrightarrow{*} v$ pour tout v.

(Remarque : l'indication donnée dans l'énoncé "interpréter α par $\emptyset$ dans le cas où $f\ 0$ diverge" était fausse, ou plutôt ne mène à rien, car si $\rho(\alpha) = \emptyset$, on ne peut plus rien dire sur les applications de f, car dans l'énoncé "pour tous $(v, v') \in A$, ou bien … ou bien …" la condition $(v, v') \in A$ est toujours fausse.)

Question 12. En s'inspirant de la question précédente, on interprète α et β par des ensembles A et B arbitraires et on exploite le fait que

$$\{\alpha \leftarrow A; \beta \leftarrow B\} \vdash a \approx a : \alpha \times \beta \rightarrow \alpha$$

Supposons que a ne diverge pas. Alors, $a \xrightarrow{*} f$, et f est telle que pour toutes valeurs $(v, v') \in A$ et $(w, w') \in B$, ou bien $f(v, w)$ et $f(v', w')$ divergent, ou bien $f(v, w) \xrightarrow{*} u$ et $f(v', w') \xrightarrow{*} u'$ et $(u, u') \in A$.

Soient alors v et w deux valeurs quelconques. On prend $A = \{(0, v)\}$ et $B = \{(0, w)\}$. Si $f(0, 0)$ diverge, alors $f(v, w)$ aussi. Si $f(0, 0)$ ne diverge pas, alors il existe u telle que $f(v, w) \xrightarrow{*} u$ et $(0, u) \in A$, c'est-à-dire $u = v$. C'est le résultat annoncé.

Question 13. Soit g une fonction totale et v une valeur du type de l'argument de g. On définit les valeurs $v_0, v_1, v_2, \ldots$ par $v_0 = v$ et pour tout i, $g\ v_i \xrightarrow{*} v_{i+1}$. On considère l'ensemble A suivant :

$$A = \{(0, v_0); (1, v_1); (2, v_2); \ldots\}$$

Remarquons que $\{\alpha \leftarrow A\} \vdash \texttt{succ} \approx g : \alpha \rightarrow \alpha$. En effet, prenons un élement quelconque de A. Il est de la forme (n, v_n) pour un certain entier n. $\texttt{succ}\ n$ se réduit en $n + 1$, et $g\ v_n$ en v_{n+1} par construction des v_i, et $(n + 1, v_{n+1}) \in A$ par construction de A.

Supposons maintenant que a se réduise en une fonction f. D'après le lemme fondamental,

$$\{\alpha \leftarrow A\} \vdash f \approx f : (\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$$

Par la remarque précédente, et le fait que $(0, v) \in A$ par construction, on a que ou bien $f\ \texttt{succ}\ 0$ et $f\ g\ v$ divergent, ou bien $f\ \texttt{succ}\ 0 \xrightarrow{*} w$ et $f\ g\ v \xrightarrow{*} u$ et $(w, u) \in A$. Dans le dernier cas, il existe forcément un entier n tel que $w = n$ et $u = v_n$. C'est le résultat annoncé, puisque v_n est le résultat de l'application $g(g(\ldots g(v)))$ itérée n fois. De plus, n ne dépend pas de v puisque n est le résultat de $f\ \texttt{succ}\ 0$.