Lambda-calcul monadique

Correction.

Documents autorisés (en particulier le poly).

Le but de ce problème est d'étudier une extension du lambda-calcul typé avec des types dits "monadiques".

La syntaxe des termes est:

$$
\begin{array}{rcll}
s, t, u, v, \ldots & ::= & x, y, z, \ldots & \text{variables} \\
& | & uv & \text{applications} \\
& | & \lambda x \cdot u & \text{abstractions} \\
& | & \texttt{let}\,\texttt{val}\,x = u\,\texttt{in}\,v & \text{séquence} \\
& | & \texttt{val}\,u & \text{conversion en valeur}
\end{array}
$$

On identifie les termes modulo α-renommage, qui est la plus petite congruence (notée $=$) telle que $\lambda x \cdot u = \lambda y \cdot (u[x := y])$ et $\texttt{let}\,\texttt{val}\,x = u\,\texttt{in}\,v = \texttt{let}\,\texttt{val}\,y = u\,\texttt{in}\,(v[x := y])$. On supposera que tous les termes sont systématiquement convertis en des termes normalisés de sorte que deux variables liées ont des noms différents, et aucune variable libre n'est liée. La variable x est *liée* dans $\texttt{let}\,\texttt{val}\,x = u\,\texttt{in}\,v$; les variables libres de $\texttt{let}\,\texttt{val}\,x = u\,\texttt{in}\,v$ sont celles de u, plus celles de v sauf x; les variables libres de $\texttt{val}\,u$ sont celles de u. Ceci nous permettra d'ignorer en grande partie les problèmes de capture de variables.

Les règles de réduction sont:

$$
\begin{array}{lrcl}
(\beta) & (\lambda x \cdot u)v & \to & u[x := v] \\
(\texttt{let}) & \texttt{let}\,\texttt{val}\,x = \texttt{val}\,u\,\texttt{in}\,v & \to & v[x := u]
\end{array}
$$

I. Le calcul non typé

1. On définit la traduction suivante du lambda-calcul monadique vers le lambda-calcul pur:

$$
\begin{array}{rclrcl}
x^\circ & = & x & (\texttt{val}\,u)^\circ & = & u^\circ \\
(uv)^\circ & = & u^\circ v^\circ & (\texttt{let}\,\texttt{val}\,x = u\,\texttt{in}\,v)^\circ & = & (\lambda x \cdot v^\circ) u^\circ \\
(\lambda x \cdot u)^\circ & = & \lambda x \cdot u^\circ & & &
\end{array}
$$

Montrer que $u^\circ[x := v^\circ] = (u[x := v])^\circ$ pour tous termes u, v et toute variable x. (On traitera les cas les plus importants.)

Par récurrence structurelle sur u. Tous les cas sont évidents: si $u = x$, $u^\circ[x := v^\circ] = x[x := v^\circ] = v^\circ = (u[x := v])^\circ$; si u est une autre variable y, $u^\circ[x := v^\circ] = y = (u[x := v])^\circ$; si u est de la forme $\lambda y \cdot u_1$, où par α-renommage on peut supposer $y \neq x$, alors $u^\circ[x := v^\circ] = (\lambda y \cdot u_1^\circ)[x := v^\circ] = \lambda y \cdot (u_1^\circ[x := v^\circ]) = \lambda y \cdot (u_1[x := v])^\circ$ (par hypothèse de récurrence) $= (\lambda y \cdot u_1[x := v])^\circ = (u[x := v])^\circ$. De même pour le `let`.

2. En déduire que si $u \to u'$ en lambda-calcul monadique, autrement dit par les règles (β) ou $(\texttt{let})$, alors $u^\circ \to u'^\circ$ dans le lambda-calcul pur.

Par récurrence sur la profondeur du rédex contracté dans u. Les cas de récurrence sont évidents. Pour les cas de base, on a:

$$
\begin{array}{rcl}
((\lambda x \cdot u)v)^\circ & = & (\lambda x \cdot u^\circ) v^\circ \\
& \to & u^\circ[x := v^\circ] = (u[x := v])^\circ
\end{array}
$$

par la question précédente.

$$\begin{array}{rcl}(\texttt{let}\,\texttt{val}\,x = \texttt{val}\,u\,\texttt{in}\,v)^\circ & = & (\lambda x \cdot v^\circ)u^\circ \\ & \to & v^\circ[x := u^\circ] = (v[x := u])^\circ\end{array}$$

par la question précédente.

3. On rappelle que le λ^*-calcul a pour termes $t, u, v, ... := x|uv|\lambda x \cdot u|(\lambda^* x \cdot u)v$, et pour règle de réduction unique $(\lambda^* \cdot u)v \to u[x := v]$. On rappelle aussi que le λ^*-calcul termine et est confluent (cf. cours). Soit $\to_{\texttt{let}}$ la notion de réduction fondée sur la seule règle $(\texttt{let})$ dans le λ-calcul monadique. Pourquoi $\to_{\texttt{let}}$ termine-t-il?

   On peut modifier la traduction $u \mapsto u^\circ$ de la question précédente de sorte que $(\texttt{let}\,\texttt{val}\,x = u\,\texttt{in}\,v)^\circ = (\lambda^* x \cdot v^\circ)u^\circ$. Alors si $u \to_{\texttt{let}} u'$, la même démonstration montre que u° se réduit en une étape dans le λ^*-calcul vers u'°. S'il y avait une réduction infinie par $\to_{\texttt{let}}$, alors il en existerait une dans le λ^*-calcul: contradiction.

4. En admettant que la notion de réduction $\to$ du λ-calcul monadique est localement confluent, pourquoi ne peut-on pas appliquer le lemme de Newman et conclure qu'elle est confluente? Justifier.

   Le lemme de Newman nécessite que $\to$ termine. Or $\to$ ne termine pas, pour la même raison que dans le lambda-calcul non monadique: par exemple $(\lambda x \cdot xx)(\lambda x \cdot xx)$ ne termine pas.

5. On définit une notion de réduction parallèle $\Rightarrow$ par:

$$\frac{}{u \Rightarrow u} \qquad \frac{u \Rightarrow u' \quad v \Rightarrow v'}{uv \Rightarrow u'v'} \qquad \frac{u \Rightarrow u'}{\lambda x \cdot u \Rightarrow \lambda x \cdot u'}$$

$$\frac{u \Rightarrow u' \quad v \Rightarrow v'}{(\lambda x \cdot u)v \Rightarrow u'[x := v']} \qquad \frac{u \Rightarrow u'}{\texttt{val}\,u \Rightarrow \texttt{val}\,u'} \qquad \frac{u \Rightarrow u' \quad v \Rightarrow v'}{\texttt{let}\,\texttt{val}\,x = \texttt{val}\,u\,\texttt{in}\,v \Rightarrow v'[x := u']}$$

   On admettra que $\Rightarrow$ est fortement confluente: si $u \Rightarrow u_1$ et $u \Rightarrow u_2$, alors il existe v tel que $u_1 \Rightarrow v$ et $u_2 \Rightarrow v$. En déduire que la notion de réduction du lambda-calcul monadique est confluente.

   C'est la même démonstration que celle faite en TD pour le lambda-calcul non monadique. Il est clair que si $u \to u'$, alors $u \Rightarrow u'$, par récurrence sur la profondeur du rédex contracté dans u. Il est aussi clair que si $u \Rightarrow u'$, alors $u \to^* u'$, par récurrence structurelle sur une preuve donnée de $u \Rightarrow u'$. On en déduit que si $u \to^* u_1$ et $u \to^* u_2$, alors $u \Rightarrow^* u_1$ et $u \Rightarrow^* u_2$. Or $\Rightarrow$, étant fortement confluente, est confluente. Donc il existe v tel que $u_1, u_2 \Rightarrow^* v$. Donc $u_1, u_2 \to^* v$.

II. Traduction du lambda-calcul par valeur

On rappelle que le lambda-calcul par valeur est le langage dont les termes sont les lambda-termes usuels (non monadiques), et dont l'unique règle de réduction est:

$$(\beta_V) \quad (\lambda x \cdot u)V \to u[x := V]$$

où V est restreint à être une *P-valeur*, c'est-à-dire une variable ou une lambda-abstraction (mais pas une application).

1. On définit la traduction des lambda-termes en lambda-termes monadiques comme suit:

$$\begin{array}{rcl} x^\star & = & \texttt{val}\,x \\ (uv)^\star & = & \texttt{let}\,\texttt{val}\,x_u = u^\star\,\texttt{in}\,(\texttt{let}\,\texttt{val}\,x_v = v^\star\,\texttt{in}\,x_u x_v) \\ (\lambda x \cdot u)^\star & = & \texttt{val}(\lambda x \cdot u^\star) \end{array}$$

   où x_u et x_v sont des variables fraîches (ni libres ni liées ni dans u ni dans v). Montrer que si $u \to u'$ par (β_V), alors $u^\star \to^+ u'^\star$ dans le lambda-calcul monadique.

C'est par récurrence sur la profondeur du rédex contracté dans u. Encore une fois, les cas de récurrence sont évidents. Reste à traiter le cas d'un (β_V)-rédex $(\lambda x \cdot u)V$. On a:

$$\begin{array}{rcl} ((\lambda x \cdot u)V)^\star & = & \texttt{let}\,\texttt{val}\,x_1 = \texttt{val}(\lambda x \cdot u^\star)\,\texttt{in}\,(\texttt{let}\,\texttt{val}\,x_2 = V^\star\,\texttt{in}\,x_1 x_2) \\ & \to & \texttt{let}\,\texttt{val}\,x_2 = V^\star\,\texttt{in}\,(\lambda x \cdot u^\star)x_2 \end{array}$$

Or comme V est une P-valeur, $V^\star$ est de la forme $\texttt{val}\,t$ pour un certain terme t. Donc

$$\begin{array}{rcl} ((\lambda x \cdot u)V)^\star & \to & \texttt{let}\,\texttt{val}\,x_2 = \texttt{val}\,t\,\texttt{in}\,(\lambda x \cdot u^\star)x_2 \\ & \to & (\lambda x \cdot u^\star)t \\ & \to & u^\star[x := t] \end{array}$$

Pour conclure, il ne reste donc plus qu'à montrer que $u^\star[x := t]$ se réduit en $(u[x := V])^\star$. C'est une deuxième récurrence structurelle sur u:

- *si $u = x$, alors $u^\star[x := t] = \texttt{val}\,t = V^\star = (u[x := V])^\star$;*
- *si u est une autre variable, alors $u^\star[x := t] = u^\star = (u[x := V])^\star$;*
- *si u est de la forme $u_1 u_2$ ou $\lambda y \cdot u_1$, c'est directement par hypothèse de récurrence.*

2. Donner un contre-exemple montrant que la proposition suivante est fausse: si $u \to u'$ dans le lambda-calcul (non nécessairement par valeur), alors $u^\star \to^* u'^\star$ dans le lambda-calcul monadique.

   On prend l'exemple le plus simple de β-rédex qui ne soit pas par valeur: $(\lambda x \cdot x)(yz)$, où y et z sont des variables. Alors $(\lambda x \cdot x)(yz) \to yz$, mais:

$$\begin{array}{rcl} ((\lambda x \cdot x)(yz))^\star & = & \texttt{let}\,\texttt{val}\,x_1 = \texttt{val}(\lambda x \cdot \texttt{val}\,x)\,\texttt{in}\,(\texttt{let}\,\texttt{val}\,x_2 = (yz)^\star\,\texttt{in}\,x_1 x_2) \\ & \to & \texttt{let}\,\texttt{val}\,x_2 = (yz)^\star\,\texttt{in}\,(\lambda x \cdot \texttt{val}\,x)x_2 \\ & \to & \texttt{let}\,\texttt{val}\,x_2 = (yz)^\star\,\texttt{in}\,\texttt{val}\,x_2 \end{array}$$

   Or $(yz)^\star = \texttt{let}\,\texttt{val}\,x_3 = \texttt{val}\,y\,\texttt{in}\,\texttt{let}\,\texttt{val}\,x_4 = \texttt{val}\,z\,\texttt{in}\,x_3 x_4 \to^+ yz$, donc $((\lambda x \cdot x)(yz))^\star$ se réduit en $\texttt{let}\,\texttt{val}\,x_2 = yz\,\texttt{in}\,\texttt{val}\,x_2$, qui est normal mais différent de la forme normale yz de $(yz)^\star$. Donc par confluence $((\lambda x \cdot x)(yz))^\star$ et $(yz)^\star$ ne sont pas convertibles, donc a fortiori pas réductibles l'un à l'autre.

III. Le calcul simplement typé

Le lambda-calcul monadique admet une discipline des types simples dans laquelle on a un nouveau constructeur de type $\boldsymbol{T}$, appelé la *monade*. Précisément, les types sont:

$$\begin{array}{rcll} F & := & b & \text{types de base} \\ & | & F \supset F & \text{types de fonction} \\ & | & \boldsymbol{T}F & \text{type monadique} \end{array}$$

Les règles de typage sont:

$$\frac{}{\Gamma, x : F \vdash x : F}\,(Ax)$$

$$\frac{\Gamma, x : F \vdash u : G}{\Gamma \vdash \lambda x \cdot u : F \supset G}\,(\supset I) \qquad \frac{\Gamma \vdash u : F \supset G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\,(\supset E)$$

$$\frac{\Gamma \vdash u : F}{\Gamma \vdash \texttt{val}\,u : \boldsymbol{T}F}\,(\boldsymbol{T}I) \qquad \frac{\Gamma \vdash u : \boldsymbol{T}F \quad \Gamma, x : F \vdash v : \boldsymbol{T}G}{\Gamma \vdash \texttt{let}\,\texttt{val}\,x = u\,\texttt{in}\,v : \boldsymbol{T}G}\,(\boldsymbol{T}E)$$

Il se trouve que l'on peut encore une fois interpréter ce système de typage comme les règles de déduction d'une logique, la *logique laxe* de Curry.

1. Montrer que si $\Gamma, x : F, \Delta \vdash u : G$ et $\Gamma \vdash v : F$ sont dérivables dans ce système, alors $\Gamma, \Delta \vdash u[x := v] : G$ est dérivable dans ce système.

   Par récurrence structurelle sur la dérivation donnée de $\Gamma, x : F, \Delta \vdash u : G$. Si $u = x$, alors $F = G$ et on peut dériver $\Gamma, \Delta \vdash u[x := v] : G$ car $u[x := v] = v$ et $F = G$, en utilisant un affaiblissement de la dérivation donnée de $\Gamma \vdash v : F$; il est clair que l'affaiblissement est admissible: on peut rajouter Δ à gauche de tous les jugements dans la dérivation de $\Gamma \vdash v : F$.

   Si u est une autre variable y, alors $y : G$ apparaît dans Γ ou dans Δ, donc $\Gamma, \Delta \vdash u : G$ est dérivable par (Ax), c'est-à-dire $\Gamma, \Delta \vdash u[x := v] : G$.

   Les cas $(\supset I)$, $(\supset E)$, $(\boldsymbol{T} I)$ sont directs par hypothèse de récurrence, de même que le cas $(\boldsymbol{T} E)$.

2. Montrer l'autoréduction: si $\Gamma \vdash u : F$ dans ce système, et $u \to u'$ par (β) ou $(\mathtt{let})$, alors $\Gamma \vdash u' : F$.

   Par récurrence structurelle sur le terme u, ou plus précisément sur la dérivation donnée de $\Gamma \vdash u : F$. Les cas de récurrence, où u n'est pas le rédex contracté, sont évidents. Dans le premier cas de base, (β), on transforme

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma, x : F \vdash u : G\end{array}}{\Gamma \vdash \lambda x \cdot u : F \supset G}(\supset I) \qquad \begin{array}{c}\vdots\\ \Gamma \vdash v : F\end{array}}{\Gamma \vdash (\lambda x \cdot u) v : G}(\supset E)$$

   en une preuve de $\Gamma \vdash u[x := v] : G$, en utilisant la question précédente avec Δ vide.

   Dans le second cas de base, $(\mathtt{let})$, on transforme

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Gamma \vdash u : F\end{array}}{\Gamma \vdash \mathtt{val}\, u : \boldsymbol{T} F}(\boldsymbol{T} I) \qquad \begin{array}{c}\vdots\\ \Gamma, x : F \vdash v : \boldsymbol{T} G\end{array}}{\Gamma \vdash \mathtt{let}\,\mathtt{val}\, x = u\ \mathtt{in}\ v : \boldsymbol{T} G}(\boldsymbol{T} E)$$

   en une preuve de $\Gamma \vdash v[x := u] : \boldsymbol{T} G$ par la question précédente, encore une fois.

3. Montrer, par un argument simple utilisant la question précédente et la question I.2, que le lambda-calcul monadique typé a la propriété de normalisation forte.

   On définit la traduction F° par: $b^\circ = b$, $(F \supset G)^\circ = F^\circ \supset G^\circ$, $(\boldsymbol{T} F)^\circ = F^\circ$. Autrement dit, F° est F avec tous les $\boldsymbol{T}$ effacés. On pose aussi $\Gamma^\circ = x_1 : F_1^\circ, \ldots, x_n : F_n^\circ$, où $\Gamma = x_1 : F_1, \ldots, x_n : F_n$. On va montrer que si $\Gamma \vdash u : F$ dans le lambda-calcul monadique, alors $\Gamma^\circ \vdash u^\circ : F^\circ$ dans le lambda-calcul simplement typé. C'est par récurrence structurelle sur la dérivation de typage donnée. Tous les cas sont évidents, sauf peut-être pour la règle $(\boldsymbol{T} E)$. (Noter que la règle $(\boldsymbol{T} I)$ disparaît purement et simplement.) Dans ce cas on a dérivé $\Gamma^\circ \vdash u^\circ : F^\circ$ et $\Gamma^\circ, x : F^\circ \vdash v^\circ : G^\circ$, donc on peut dériver $\Gamma^\circ \vdash (\lambda x \cdot v^\circ) u^\circ : G^\circ$; on conclut car $(\lambda x \cdot v^\circ) u^\circ = (\mathtt{let}\,\mathtt{val}\, x = u\ \mathtt{in}\ v)^\circ$ et $G^\circ = (\boldsymbol{T} G)^\circ$.

   En particulier, si u est un lambda-terme monadique typé, u° est un lambda-terme typé, qui est donc fortement normalisant. Par la question I.2, si u n'était pas fortement normalisant, u° aurait une réduction infinie, ce qui est impossible.

4. En déduire que la logique laxe est cohérente.

   Montrons qu'il n'y a pas de lambda-terme monadique u tel que $\vdash u : b$ soit dérivable, où b est un type de base. Par la question précédente, u peut être supposé normal. u ne peut pas être une variable car il n'y a rien à gauche de $\vdash$, ne peut pas être une λ, une $\mathtt{val}$, ou un $\mathtt{let}$. Donc u est une application $u'u''$, et u' est d'un type fonctionnel ($\supset$). Écrivons u plus généralement sous la forme $u_0 u_1 \ldots u_n$, où $n \geq 1$ et u_0 n'est pas une application. Encore une fois, u_0 ne peut pas être une variable car il n'y a pas de variable à gauche de $\vdash$. u_0 ne peut pas être une λ-abstraction, car u est normal. De plus, comme $n \geq 1$, u_0 est d'un certain type $\supset$, donc par typage u_0 ne peut être ni une $\mathtt{val}$ ni un $\mathtt{let}$. Donc u_0 n'existe pas, en conséquence u non plus.

5. On définit une traduction de la logique minimale intuitionniste en logique laxe par:

$$b^\star = b \qquad (F \supset G)^\star = F^\star \supset \boldsymbol{T} G^\star$$

On étend cette traduction aux contextes par $(x_1 : F_1, \ldots, x_n : F_n)^\star = x_1 : F_1^\star, \ldots, x_n : F_n^\star$. Montrer que si u est un lambda-terme (non monadique) tel que $\Gamma \vdash u : F$, alors $\Gamma^\star \vdash u^\star : \boldsymbol{T} F^\star$. ($u^\star$ est défini dans la partie II.)

Par récurrence structurelle sur u, ou plutôt sur la dérivation donnée de $\Gamma \vdash u : F$. Si u est une variable x, $u^\star = \mathtt{val}\, x$ est bien de type $\boldsymbol{T}F^*$ dans $\Gamma^\star$ par la règle $(\boldsymbol{T}I)$.
Si u est une application $u_1 u_2$, avec par hypothèse de récurrence des dérivations de $\Gamma^\star \vdash u_1^\star : \boldsymbol{T}(F^\star \supset \boldsymbol{T}G^\star)$ (u_1 étant de type $F \supset G$) et de $\Gamma^\star \vdash u_2^\star : \boldsymbol{T}F^\star$ (u_2 étant de type F), on dérive:

$$\dfrac{\dfrac{\vdots}{\Gamma^\star \vdash u_1^\star : \boldsymbol{T}(F^\star \supset \boldsymbol{T}G^\star)} \qquad \dfrac{\dfrac{\vdots}{\Gamma^\star, x_1 : F^\star \supset \boldsymbol{T}G^\star \vdash u_2^\star : \boldsymbol{T}F^\star} \quad \dfrac{}{\Gamma^\star, x_1 : F^\star \supset \boldsymbol{T}G^\star, x_2 : F^\star \vdash x_1 x_2 : \boldsymbol{T}G^\star}\,(Ax), (\supset E)}{\Gamma^\star, x_1 : F^\star \supset \boldsymbol{T}G^\star \vdash \mathtt{let\,val}\, x_2 = u_2^\star \,\mathtt{in}\, x_1 x_2 : \boldsymbol{T}G^\star}\,(\boldsymbol{T}E)}{\Gamma^\star \vdash \mathtt{let\,val}\, x_1 = u_1^\star \,\mathtt{in}\, (\mathtt{let\,val}\, x_2 = u_2^\star \,\mathtt{in}\, x_1 x_2) : \boldsymbol{T}G^\star}\,(\boldsymbol{T}E)$$

où l'on a appliqué l'affaiblissement pour obtenir la preuve de $\Gamma^\star, x_1 : F^\star \supset \boldsymbol{T}G^\star \vdash u_2^\star : \boldsymbol{T}F^\star$. Et bien sûr ceci est une dérivation de $\Gamma^\star \vdash u^\star : \boldsymbol{T}G^\star$.
Si u est une abstraction $\lambda x \cdot v$, on a par hypothèse de récurrence une dérivation de $\Gamma^\star, x : F^\star \vdash v^\star : \boldsymbol{T}G^\star$ (à partir de $\Gamma, x : F \vdash v : G$), d'où l'on déduit:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma^\star, x : F^\star \vdash v^\star : \boldsymbol{T}G^\star}}{\Gamma^\star \vdash \lambda x \cdot v^\star : F^\star \supset \boldsymbol{T}G^\star}\,(\supset I)}{\Gamma^\star \vdash \mathtt{val}(\lambda x \cdot v^\star) : \boldsymbol{T}(F^\star \supset \boldsymbol{T}G^\star)}\,(\mathtt{val}\, I)$$

qui est une dérivation de $\Gamma^\star \vdash u^\star : (F \supset G)^\star$.

IV. La monade des continuations

On se fixe un type quelconque F_0 une fois pour toutes. On pose $\neg_0 F = F \supset F_0$, et l'on définit les constructions suivantes dans le lambda-calcul typé (non monadique):

$$\begin{array}{rcl} \boldsymbol{T}F & = & \neg_0 \neg_0 F \\ \mathtt{val}\, u & = & \lambda z \cdot zu \\ \mathtt{let\,val}\, x = u \,\mathtt{in}\, v & = & \lambda z \cdot u(\lambda x \cdot vz) \end{array}$$

où z est une variable fraîche.

1. Montrer que, avec ces définitions, et en supposant que l'on dispose de la règle η du lambda-calcul, les règles de réduction (β) et $(\mathtt{let})$ de la section I sont valides.

   Le cas de base (β) est évident, quant au cas $(\mathtt{let})$, on a:

   $$\begin{array}{rcl} \mathtt{let\,val}\, x = \mathtt{val}\, u \,\mathtt{in}\, v & = & \lambda z \cdot (\lambda z' \cdot z' u)(\lambda x \cdot vz) \\ & \to & \lambda z \cdot (\lambda x \cdot vz) u \\ & \to & \lambda z \cdot (vz)[x := u] \\ & = & \lambda z \cdot v[x := u] z \end{array}$$

   qui se réduit par η en $v[x := u]$.
   Les cas de récurrence sont évidents.

2. Montrer que, avec ces définitions, les règles de typage de la section III sont valides.

Il s'agit de vérifier les règles ($\boldsymbol{T}I$) et ($\boldsymbol{T}E$). Pour ($\boldsymbol{T}I$):

$$\dfrac{\dfrac{\dfrac{}{\Gamma, z : \neg_0 F \vdash z : \neg_0 F} \qquad \dfrac{\begin{matrix}\vdots \\ \Gamma \vdash u : F\end{matrix}}{\Gamma, z : \neg_0 F \vdash u : F}\,\textit{(affaiblissement)}}{\Gamma, z : \neg_0 F \vdash zu : F_0}\,(\supset E)}{\Gamma \vdash \lambda z \cdot zu = \mathtt{val}\, u : \neg_0\neg_0 F = \boldsymbol{T} F}$$

Pour ($\boldsymbol{T}E$):

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots \\ \Gamma \vdash u : \neg_0\neg_0 F\end{matrix}}{\Gamma, z : \neg_0 G \vdash u : \neg_0\neg_0 F}\,\textit{(aff.)} \qquad \dfrac{\dfrac{\dfrac{\begin{matrix}\vdots \\ \Gamma, x : F \vdash v : \neg_0\neg_0 G\end{matrix}}{\Gamma, x : F, z : \neg_0 G \vdash v : \neg_0\neg_0 G}\,\textit{(aff.)} \qquad \dfrac{}{\Gamma, x : F, z : \neg_0 G \vdash z : \neg_0 G}\,(Ax)}{\Gamma, x : F, z : \neg_0 G \vdash vz : F_0}\,(\supset E)}{\Gamma, z : \neg_0 G \vdash \lambda x \cdot vz : \neg_0 F}\,(\supset I)}{\Gamma, z : \neg_0 G \vdash u(\lambda x \cdot vz) : F_0}\,(\supset E)}{\Gamma \vdash \lambda z \cdot u(\lambda x \cdot vz) : \neg_0\neg_0 G}\,(\supset I)$$