Programmation fonctionnelle et systèmes de types (MPRI 2-4-2)
Examen partiel 2009–2010

François Pottier

17 novembre 2009

Ce problème étudie la question de la sûreté du typage dans le système F_η. Il est divisé en plusieurs parties. Les parties 1 à 3 sont introductives. Les parties 4 et 5 sont indépendantes l'une de l'autre. La partie 6 fait suite à la partie 5. La partie 7 ne dépend que de la partie 1.

1 Types et coercions

On rappelle la syntaxe des types commune au système F et au système F_η :

$$T ::= X \mid T \to T \mid \forall X.T$$

Les types du système F_η sont reliés par une relation dite d'instance, ou de sous-typage, notée $\leq$. Cette relation est définie par un jeu de règles relativement nombreuses : un jugement de la forme $T_1 \leq T_2$ peut admettre une dérivation complexe. Pour faciliter le raisonnement à propos de ces dérivations, nous introduisons un langage de *coercions*. Une coercion c peut être considérée comme le témoin, ou la preuve, d'une affirmation de la forme $T_1 \leq T_2$. La syntaxe des coercions est la suivante :

$$c ::= \mathit{id} \mid c; c \mid \mathit{intro} \mid @T \mid c \to c \mid \forall X.c \mid \mathit{distrib}$$

On notera qu'une coercion peut mentionner un type (via la forme $@T$), donc peut avoir des variables de types libres. On notera que la forme $\forall X.c$ lie la variable de types X dans la coercion c.

La syntaxe ci-dessus peut sembler mystérieuse, mais la signification de chacune de ces constructions doit s'éclairer lorsque l'on considère les règles de preuve suivantes. Ces règles définissent un jugement $c : T_1 \leq T_2$, que l'on peut lire, par exemple : *la coercion c démontre que T_2 est instance de T_1*. On pourra lire également : *la coercion c convertit une valeur de type T_1 en une valeur de type T_2*.

Réflexivité
$$\mathit{id} : T \leq T$$

Transitivité
$$\frac{c_1 : T_1 \leq T_2 \qquad c_2 : T_2 \leq T_3}{(c_1; c_2) : T_1 \leq T_3}$$

$\forall$-Introduction
$$\frac{X \# T}{\mathit{intro} : T \leq \forall X.T}$$

$\forall$-Elimination
$$@T_2 : \forall X.T_1 \leq [X \mapsto T_2]T_1$$

$\to$-Congruence
$$\frac{c_1 : T_1' \leq T_1 \qquad c_2 : T_2 \leq T_2'}{c_1 \to c_2 : T_1 \to T_2 \leq T_1' \to T_2'}$$

$\forall$-Congruence
$$\frac{c : T_1 \leq T_2}{\forall X.c : \forall X.T_1 \leq \forall X.T_2}$$

Distribution
$$\mathit{distrib} : \forall X.(T_1 \to T_2) \leq (\forall X.T_1) \to (\forall X.T_2)$$

Cette présentation n'est pas exactement celle considérée en cours. Toutefois, les exemples suivants devraient vous convaincre (et on pourrait démontrer) que la relation définie ici est bien identique à celle considérée en cours.

Question 1 *Donner un témoin c de la relation $T_1 \to T_2 \leq (\forall X.X) \to T_2$. On donnera la dérivation du jugement $c : T_1 \to T_2 \leq (\forall X.X) \to T_2$.* ◇

Solution. Une réponse est $(@T_1) \to \mathit{id}$. La dérivation est :

$$\frac{\dfrac{}{@T_1 : \forall X.X \leq T_1}\ \forall\text{-Elimination} \qquad \dfrac{}{\mathit{id} : T_2 \leq T_2}\ \text{Réflexivité}}{(@T_1) \to \mathit{id} : T_1 \to T_2 \leq (\forall X.X) \to T_2}\ \to\text{-Congruence}$$

Question 2 *Donner un témoin de la relation* $\forall X.\forall Y.(T_1 \to T_2) \leq (\forall X.\forall Y.T_1) \to (\forall X.\forall Y.T_2)$. *On ne demande pas de dérivation.* ◇

Solution. Une réponse est $(\forall X.\mathit{distrib}); \mathit{distrib}$. □

Question 3 *Donner un témoin de la relation* $\forall X.\forall Y.(X \to Y) \leq T_1 \to T_2$. *On ne demande pas de dérivation.*◇

Solution. Une réponse est $@T_1; @T_2$. □

Question 4 *Donner un témoin de la relation* $\forall X.(X \to X) \leq \forall Y.\forall Z.((Y \to Z) \to (Y \to Z))$. *On ne demande pas de dérivation.* ◇

Solution. Une réponse est $\mathit{intro}; \mathit{intro}; \forall Y.\forall Z.@(Y \to Z)$. □

2 Typage

On donne maintenant la syntaxe des termes du système F_η :

$$t ::= x \mid \lambda x.t \mid t\ t \mid \Lambda X.t \mid c\ t$$

Cette syntaxe est explicitement typée, au sens où Λ-abstractions et coercions sont explicites. Toutefois, pour des raisons de légèreté, les λ-abstractions ne portent pas explicitement le type de leur argument. (Ce choix est quelque peu non standard, mais correspond à ce qui a été présenté en cours.) La construction $c\ t$ doit être lue comme l'application d'une coercion à un terme. On notera que, puisqu'une coercion peut avoir des variables de types libres, il en va de même d'un terme.

Les règles de typage du système F_η sont les suivantes. (Un environnement de typage Γ est une séquence de liaisons de la forme $x : T$.) L'application d'une coercion à un terme, via la règle Sub, permet d'exploiter la relation d'instance. Ainsi, cette présentation du système F_η est dirigée par la syntaxe, ce qui facilitera la preuve de sûreté du typage.

$$\textsc{Var}\ \Gamma \vdash x : \Gamma(x) \qquad \textsc{Abs}\ \frac{\Gamma; x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda x.t : T_1 \to T_2} \qquad \textsc{App}\ \frac{\Gamma \vdash t_1 : T_1 \to T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : T_2}$$

$$\textsc{TAbs}\ \frac{\Gamma \vdash t : T \qquad X \# \Gamma}{\Gamma \vdash \Lambda X.t : \forall X.T} \qquad \textsc{Sub}\ \frac{\Gamma \vdash t : T_1 \qquad c : T_1 \leq T_2}{\Gamma \vdash c\ t : T_2}$$

Question 5 *Pourquoi l'application de types* $t ::= \ldots \mid t\ T$, *qui dans le système F permet d'éliminer un quantificateur universel, n'apparaît-elle pas ici ? Par quelle construction est-elle remplacée ?* ◇

Solution. Elle est remplacée par l'application d'une coercion : $t\ T$ s'écrit ici $(@T)\ t$. □

3 Sémantique

On dote à présent les termes (dont la syntaxe a été donnée ci-dessus) d'une sémantique. Il s'agit d'une sémantique à réduction à petits pas, en appel par valeur.

Pour définir cette sémantique, on spécifie d'abord la syntaxe des valeurs et des contextes d'évaluation :

$$v ::= \lambda x.t \mid \Lambda X.v$$
$$E ::= [] \mid E\ t \mid v\ E \mid \Lambda X.E \mid c\ E$$

Ensuite, la relation de réduction $\longrightarrow$ est définie, de façon inductive, par les règles suivantes :

$$(\lambda x.t)\ v \longrightarrow [x \mapsto v]t \tag{1}$$
$$id\ v \longrightarrow v \tag{2}$$
$$(c_1; c_2)\ v \longrightarrow c_2\ (c_1\ v) \tag{3}$$
$$intro\ v \longrightarrow \Lambda X.v \qquad \text{si } X \# v \tag{4}$$
$$(@T)\ (\Lambda X.v) \longrightarrow [X \mapsto T]v \tag{5}$$
$$(c_1 \to c_2)\ (\lambda x.t) \longrightarrow \lambda x.(c_2\ ([x \mapsto c_1\ x]t)) \tag{6}$$
$$(\forall X.c)\ (\Lambda X.v) \longrightarrow \Lambda X.(c\ v) \tag{7}$$
$$distrib\ (\Lambda X.\lambda x.t) \longrightarrow \lambda x.\Lambda X.([x \mapsto @X\ x]t) \tag{8}$$
$$E[t_1] \longrightarrow E[t_2] \qquad \text{si } t_1 \longrightarrow t_2 \tag{9}$$

Cette sémantique est « à effacement des types », au sens où, même si les termes contiennent des annotations liées au typage (Λ-abstractions, coercions), et même si les règles de réduction ci-dessus suggèrent que ces annotations jouent un rôle au cours de calcul, ceci n'est pas réellement le cas. On peut en fait effacer ces annotations avant l'exécution sans affecter le résultat du calcul. On pourrait démontrer ce fait en établissant que la fonction d'effacement des annotations est une simulation. Nous ne le ferons pas ici.

4 Relation entre les systèmes F et F_η

On souhaite maintenant préciser la relation entre les systèmes F et F_η.
On rappelle la syntaxe des termes du système F :

$$u ::= x \mid \lambda x.u \mid u\ u \mid \Lambda X.u \mid u\ T$$

Si u est un terme du système F, on note $\lfloor u \rfloor$ le λ-terme pur obtenu à partir de u par effacement de toutes les abstractions et applications de types.

Dans le système F, on ne dispose ni de coercions, ni de la relation d'instance du système F_η : on ne dispose que des constructions classiques d'abstraction et d'application de types. Cependant, une traduction du système F_η dans le système F est possible. La traduction des constructions communes aux systèmes F et F_η est bien sûr triviale : la seule difficulté est de traduire l'application d'une coercion à un terme. Nous considérons à présent cette question.

Question 6 *Pour toute coercion $c : T_1 \leq T_2$, et pour tout terme u du système F tel que $\Gamma \vdash u : T_1$ est dérivable dans le système F, construire un terme du système F, que l'on notera $[\![c\ u]\!]$, de sorte que les deux propriétés suivantes soient satisfaites :*

1. *$\Gamma \vdash [\![c\ u]\!] : T_2$ est dérivable dans le système F ;*
2. *les λ-termes purs $\lfloor u \rfloor$ et $\lfloor [\![c\ u]\!] \rfloor$ sont η-équivalents.*

On se contentera de définir la fonction $[\![\cdot\ \cdot]\!]$. On ne démontrera pas explicitement que ces deux propriétés sont satisfaites. ◇

Solution. La traduction des coercions vers le système F se définit par induction sur la structure des coercions :

$$\begin{aligned}
[\![id\ u]\!] &= u \\
[\![(c_1; c_2)\ u]\!] &= [\![c_2\ [\![c_1\ u]\!]]\!] \\
[\![intro\ u]\!] &= \Lambda X.u \\
[\![@T\ u]\!] &= u\ T \\
[\![(c_1 \to c_2)\ u]\!] &= \lambda x.[\![c_2\ (u\ [\![c_1\ x]\!])]\!] \\
[\![(\forall X.c)\ u]\!] &= \Lambda X.[\![c\ (u\ X)]\!] \\
[\![distrib\ u]\!] &= \lambda x.\Lambda X.u\ X\ (x\ X)
\end{aligned}$$

(Les variables x et X sont choisies fraîches lorsque cela est possible/nécessaire, de façon à éviter toute capture.) Le fait que $[\![c\ u]\!]$ admet le type T_2 si $c : T_1 \leq T_2$ et si u admet le type T_1 se vérifie par induction sur c. Le fait que $\lfloor [\![c\ u]\!] \rfloor$ est η-équivalent à $\lfloor u \rfloor$ se vérifie également par induction sur c. Ces démonstrations sont omises.□

Il découle de ce résultat que, si $\Gamma \vdash t : T$ est dérivable dans le système F_η, alors il existe un terme u tel que $\lfloor t \rfloor$ et $\lfloor u \rfloor$ sont η-équivalents et tel que $\Gamma \vdash u : T$ est dérivable dans le système F.

En d'autres termes, le système F_η n'est pas beaucoup plus expressif que le système F : il permet simplement d'omettre (ou, plus précisément, de remplacer par des coercions) certaines η-expansions explicites. On peut ainsi espérer gagner un peu en concision (si les coercions ne sont pas données explicitement) et surtout en efficacité (car les coercions sont effacées avant l'exécution, tandis qu'un η-redex pourra donner lieu à du code).

Mitchell [1] a démontré la réciproque de ce résultat, ce qui implique que le système F_η est exactement la « clôture du système F modulo η. »

5 Sûreté du système F_η

On démontre maintenant que la réduction préserve le typage. Comme dans le cours, la preuve comporte un cas par règle de réduction, donc neuf cas en tout. On ne demande de traiter ici **qu'une partie** de ces cas.

Question 7 (Auto-réduction) *Démontrer que, si $\Gamma \vdash t : T$, et si t se réduit en t'* ***via l'une des règles de réduction 4 à 8,*** *alors $\Gamma \vdash t' : T$. Dans chaque cas, on analysera la forme de la dérivation du jugement $\Gamma \vdash t : T$, et on construira une dérivation du jugement $\Gamma \vdash t' : T$. On admettra sans démonstration les lemmes auxiliaires standard (substitution, etc.). On prendra toutefois soin de donner avec précision les énoncés de ces lemmes.* ◇

Solution. Bien que seuls les cas des règles 4 à 8 aient été demandés, ce corrigé traite l'ensemble des cas, de 1 à 9. Voici d'abord les énoncés des quatre lemmes dont nous aurons besoin :

1. Substitution des types :

   $\Gamma \vdash t : T_2$ implique $[X \mapsto T_1]\Gamma \vdash [X \mapsto T_1]t : [X \mapsto T_1]T_2$.

   Notons que cet énoncé exige une substitution dans les trois composantes du jugement de typage, puisque toutes trois peuvent contenir des variables de types libres.

2. Substitution des termes :

   $\Gamma; x : T_1 \vdash t_2 : T_2$ et $\Gamma \vdash t_1 : T_1$ impliquent $\Gamma \vdash [x \mapsto t_1]t_2 : T_2$.

   Notons que, pour démontrer ce lemme par induction, il faut en généraliser l'énoncé :

   $\Gamma_1; x : T_1; \Gamma_2 \vdash t_2 : T_2$ et $\Gamma_1 \vdash t_1 : T_1$ impliquent $\Gamma_1; \Gamma_2 \vdash [x \mapsto t_1]t_2 : T_2$.

3. Affaiblissement :

   $\Gamma_1; \Gamma_2 \vdash t : T$ et $x \notin \mathrm{fv}(t) \setminus \mathrm{dom}(\Gamma_2)$ impliquent $\Gamma_1; x : T'; \Gamma_2 \vdash t : T$.

4. Compositionalité :

   si $\Gamma \vdash E[t] : T$, alors il existe un type T' tel que $\Gamma \vdash t : T'$ et tel que, pour tout t', $\Gamma \vdash t' : T'$ implique $\Gamma \vdash E[t'] : T'$.

La démonstration du théorème de préservation du typage se fait par induction sur la dérivation du jugement $t \longrightarrow t'$. Il y a donc autant de cas que de règles de réduction. Dans chaque cas, on tire parti du fait que le système de types est dirigé par la syntaxe pour déconstruire l'hypothèse $\Gamma \vdash t : T$ sans introduire d'analyse de cas supplémentaire.

◦ *Cas* $(\lambda x.t)\ v \longrightarrow [x \mapsto v]t$. La dérivation de typage est nécessairement de la forme :

$$\dfrac{\dfrac{\Gamma; x : T_1 \vdash t : T_2 \quad (\mathbf{2})}{\Gamma \vdash \lambda x.t : T_1 \to T_2} \qquad \Gamma \vdash v : T_1 \quad (\mathbf{1})}{\Gamma \vdash (\lambda x.t)\ v : T_2}$$

Le lemme de substitution des termes, appliqué à (2) et (1), donne $\Gamma \vdash [x \mapsto v]t : T_2$, qui est précisément le résultat attendu. On notera que ce raisonnement est identique à celui effectué en cours pour le λ-calcul simplement typé. La présence de la règle de sous-typage du système F_η ne complique pas les choses, parce que, grâce à l'introduction de coercions explicites, cette règle est dirigée par la syntaxe. Bien sûr, il nous reste maintenant à montrer que les règles de réduction des coercions préservent le typage. En organisant la preuve de cette manière, nous en maîtrisons mieux la complexité.

◦ *Cas* $id\ v \longrightarrow v$. La dérivation de typage est nécessairement de la forme :

$$\frac{\Gamma \vdash v : T \quad (\mathbf{1}) \qquad id : T \leq T}{\Gamma \vdash id\ v : T}$$

La prémisse (1) est exactement le résultat attendu.

◦ *Cas* $(c_1; c_2)\ v \longrightarrow c_2\ (c_1\ v)$. La dérivation de typage est nécessairement de la forme :

$$\frac{\Gamma \vdash v : T_1 \qquad \dfrac{c_1 : T_1 \leq T_2 \qquad c_2 : T_2 \leq T_3}{(c_1; c_2) : T_1 \leq T_3}}{\Gamma \vdash (c_1; c_2)\ v : T_3}$$

On construit alors la dérivation suivante :

$$\frac{\dfrac{\Gamma \vdash v : T_1 \qquad c_1 : T_1 \leq T_2}{\Gamma \vdash c_1\ v : T_2} \qquad c_2 : T_2 \leq T_3}{\Gamma \vdash c_2\ (c_1\ v) : T_3}$$

qui fournit le résultat attendu.

◦ *Cas intro* $v \longrightarrow \Lambda X.v$, où $X \# v$. La dérivation de typage est nécessairement de la forme :

$$\frac{\Gamma \vdash v : T \qquad \dfrac{Y \# T}{intro : T \leq \forall Y.T}}{\Gamma \vdash intro\ v : \forall Y.T}$$

L'hypothèse $X \# v$ implique que, pour toute variable $X' \# v$, les valeurs $\Lambda X.v$ et $\Lambda X'.v$ sont égales. De même, l'hypothèse $Y \# T$ implique que, pour toute variable $Y' \# T$, les types $\forall Y.T$ et $\forall Y'.T$ sont égaux. En d'autres termes, nous pouvons renommer X et Y à volonté, tant que nous restons hors du support de v et T. Nous allons les renommer tous deux en une même variable Z, que nous choisissons telle que $Z \# \Gamma, v, T$. Nous avons donc :

$$\frac{\Gamma \vdash v : T \qquad \dfrac{Z \# T}{intro : T \leq \forall Z.T}}{\Gamma \vdash intro\ v : \forall Z.T}$$

On construit alors la dérivation suivante :

$$\frac{\Gamma \vdash v : T \qquad Z \# \Gamma}{\Gamma \vdash \Lambda Z.v : \forall Z.T}$$

qui fournit le résultat attendu.

◦ *Cas* $(@T)\ (\Lambda X.v) \longrightarrow [X \mapsto T]v$. La dérivation de typage est nécessairement de la forme :

$$\frac{\dfrac{\Gamma \vdash v : T_1 \quad (\mathbf{2}) \qquad X \# \Gamma \quad (\mathbf{1})}{\Gamma \vdash \Lambda X.v : \forall X.T_1} \qquad @T : \forall X.T_1 \leq [X \mapsto T]T_1}{\Gamma \vdash (@T)\ (\Lambda X.v) : [X \mapsto T]T_1}$$

Le lemme de substitution des types, appliqué à la prémisse (2), donne $[X \mapsto T]\Gamma \vdash [X \mapsto T]v : [X \mapsto T]T_1$. Grâce à l'hypothèse (1), ceci s'écrit en fait $\Gamma \vdash [X \mapsto T]v : [X \mapsto T]T_1$, qui est le résultat attendu.

◦ *Cas* $(c_1 \to c_2)\ (\lambda x.t) \longrightarrow \lambda x.(c_2\ ([x \mapsto c_1\ x]t))$. La dérivation de typage est nécessairement de la forme :

$$\frac{\dfrac{\Gamma; x : T_1 \vdash t : T_2 \quad (\mathbf{1})}{\Gamma \vdash \lambda x.t : T_1 \to T_2} \qquad \dfrac{c_1 : T_1' \leq T_1 \qquad c_2 : T_2 \leq T_2'}{c_1 \to c_2 : T_1 \to T_2 \leq T_1' \to T_2'}}{\Gamma \vdash (c_1 \to c_2)\ (\lambda x.t) : T_1' \to T_2'}$$

Construisons la petite dérivation suivante :

$$\frac{\Gamma; x : T_1' \vdash x : T_1' \qquad c_1 : T_1' \leq T_1}{\Gamma; x : T_1' \vdash c_1\ x : T_1 \quad (\mathbf{2})}$$

D'après le lemme d'affaiblissement, la prémisse (1) peut s'écrire $\Gamma; x : T_1'; x : T_1 \vdash t : T_2$ (**3**), la seconde liaison de x dans l'environnement venant masquer la première. Nous pouvons maintenant appliquer le lemme de substitution des termes aux faits (3) et (2), ce qui donne $\Gamma; x : T_1' \vdash [x \mapsto c_1\ x]t : T_2$ (**4**). Nous concluons en construisant la dérivation suivante :

$$\frac{\dfrac{\Gamma; x : T_1' \vdash [x \mapsto c_1\ x]t : T_2 \quad (4) \qquad c_2 : T_2 \leq T_2'}{\Gamma; x : T_1' \vdash c_2\ ([x \mapsto c_1\ x]t) : T_2'}}{\Gamma \vdash \lambda x.c_2\ ([x \mapsto c_1\ x]t) : T_1' \to T_2'}$$

qui fournit le résultat attendu.

◦ *Cas* $(\forall X.c)\ (\Lambda X.v) \longrightarrow \Lambda X.(c\ v)$. La dérivation de typage est nécessairement de la forme :

$$\frac{\dfrac{\Gamma \vdash v : T_1 \qquad X \# \Gamma}{\Gamma \vdash \Lambda X.v : \forall X.T_1} \qquad \dfrac{c : T_1 \leq T_2}{\forall X.c : \forall X.T_1 \leq \forall X.T_2}}{\Gamma \vdash (\forall X.c)\ (\Lambda X.v) : \forall X.T_2}$$

On construit alors la dérivation suivante :

$$\frac{\dfrac{\Gamma \vdash v : T_1 \qquad c : T_1 \leq T_2}{\Gamma \vdash c\ v : T_2} \qquad X \# \Gamma}{\Gamma \vdash \Lambda X.(c\ v) : \forall X.T_2}$$

qui fournit le résultat attendu.

◦ *Cas distrib* $(\Lambda X.\lambda x.t) \longrightarrow \lambda x.\Lambda X.([x \mapsto @X\ x]t)$. La dérivation de typage est nécessairement de la forme :

$$\frac{\dfrac{\dfrac{\Gamma; x : T_1 \vdash t : T_2 \quad (\mathbf{1})}{\Gamma \vdash \lambda x.t : T_1 \to T_2} \qquad X \# \Gamma}{\Gamma \vdash \Lambda X.\lambda x.t : \forall X.(T_1 \to T_2)} \qquad \mathit{distrib} : \forall X.(T_1 \to T_2) \leq (\forall X.T_1) \to (\forall X.T_2)}{\Gamma \vdash \mathit{distrib}\ (\Lambda X.\lambda x.t) : (\forall X.T_1) \to (\forall X.T_2)}$$

Construisons la petite dérivation suivante :

$$\frac{\Gamma; x : \forall X.T_1 \vdash x : \forall X.T_1 \qquad @X : \forall X.T_1 \leq T_1}{\Gamma; x : \forall X.T_1 \vdash @X\ x : T_1 \quad (\mathbf{2})}$$

D'après le lemme d'affaiblissement, la prémisse (1) peut s'écrire $\Gamma; x : \forall X.T_1; x : T_1 \vdash t : T_2$ (**3**), la seconde liaison de x dans l'environnement venant masquer la première. Nous pouvons maintenant appliquer le lemme de substitution des termes aux faits (3) et (2), ce qui donne $\Gamma; x : \forall X.T_1 \vdash [x \mapsto @X\ x]t : T_2$ (**4**). Nous concluons en construisant la dérivation suivante :

$$\frac{\dfrac{\Gamma; x : \forall X.T_1 \vdash [x \mapsto @X\ x]t : T_2 \quad (4) \qquad X \# (\Gamma; x : \forall X.T_1)}{\Gamma; x : \forall X.T_1 \vdash \Lambda X.([x \mapsto @X\ x]t) : \forall X.T_2}}{\Gamma \vdash \lambda x.\Lambda X.([x \mapsto @X\ x]t) : (\forall X.T_1) \to (\forall X.T_2)}$$

qui fournit le résultat attendu.

◦ *Cas* $E[t_1] \longrightarrow E[t_2]$, où $t_1 \longrightarrow t_2$ (**1**). La dérivation de typage est nécessairement de la forme :

$$\frac{\begin{array}{c}\Gamma \vdash t_1 : T' \quad (\mathbf{2}) \\ \vdots \end{array}}{\Gamma \vdash E[t_1] : T}$$

Par hypothèse d'induction, (1) et (2) impliquent $\Gamma \vdash t_2 : T'$ (**3**). Nous pouvons alors reconstruire une dérivation analogue à la précédente :

$$\dfrac{\begin{array}{c}\Gamma \vdash t_2 : T' \quad (3)\\ \vdots\end{array}}{\Gamma \vdash E[t_2] : T}$$

qui fournit le résultat attendu. Le raisonnement informel que nous venons d'effectuer correspond à une application du lemme de compositionalité. □

On démontre ensuite la propriété de progrès. Comme dans le cours, la preuve comporte un cas par construction dans la syntaxe des termes du système F_η, donc cinq cas en tout. On ne demande de traiter ici **qu'une partie** de ces cas.

Question 8 (Progrès) *Démontrer qu'un terme* ***de la forme*** $\Lambda X.t$ ***ou*** $c\ t$*, bien typé dans l'environnement vide, soit se réduit soit est une valeur. On admettra sans démonstration le lemme de classification des valeurs, dont on donnera avec précision l'énoncé.* ◇

Solution. Le lemme de classification des valeurs est le suivant :

- si une valeur bien typée admet un type flèche, alors elle est de la forme $\lambda x.t$;
- si une valeur bien typée admet un type universel, alors elle est de la forme $\Lambda X.v$.

La preuve de la propriété de progrès se fait par induction sur la structure d'un terme t tel que $\vdash t : T$. Bien que seuls les cas $\Lambda X.t$ et $c\ t$ aient été demandés, ce corrigé traite l'ensemble des cas.

∘ *Cas* x. Une variable ne peut être bien typée (dans l'environnement vide) : contradiction.

∘ *Cas* $\lambda x.t$. Ce terme est une valeur.

∘ *Cas* $t_1\ t_2$. Du fait que $t_1\ t_2$ est bien typé, on déduit que t_1 et t_2 le sont également. Il s'ensuit, par hypothèse d'induction, que chacun de ces termes soit se réduit, soit est une valeur. On raisonne alors par cas, comme en cours :

- si t_1 se réduit, alors, parce que $[]\ t_2$ est un contexte d'évaluation, $t_1\ t_2$ se réduit ;
- si t_1 est une valeur v_1 et si t_2 se réduit, alors, parce que $v_1\ []$ est un contexte d'évaluation, $v_1\ t_2$ se réduit ;
- si t_1 et t_2 sont des valeurs v_1 et v_2, alors, du fait que $v_1\ v_2$ est bien typé, on déduit que v_1 admet un type flèche ; il en découle, via le lemme de classification des valeurs, que v_1 est une λ-abstraction ; le terme $v_1\ v_2$ est donc un β-redex, et se réduit.

∘ *Cas* $\Lambda X.t$. Du fait que $\Lambda X.t$ est bien typé, on déduit que t l'est également, donc, par hypothèse d'induction, soit se réduit, soit est une valeur v. Dans le premier cas, parce que $\Lambda X.[]$ est un contexte d'évaluation, $\Lambda X.t$ se réduit. Dans le second cas, $\Lambda X.t$ est une valeur $\Lambda X.v$.

∘ *Cas* $c\ t$. Du fait que $c\ t$ est bien typé, on déduit que t l'est également, donc, par hypothèse d'induction, soit se réduit, soit est une valeur v. Dans le premier cas, parce que $c\ []$ est un contexte d'évaluation, $c\ t$ se réduit. Dans le second cas, le terme qui nous intéresse est de la forme $c\ v$. Nous raisonnons alors par cas suivant la forme de la coercion c.

Sous-cas *id*. Le terme *id* v se réduit en v.

Sous-cas $(c_1; c_2)$. Le terme $(c_1; c_2)\ v$ se réduit en $c_2\ (c_1\ v)$.

Sous-cas *intro*. Le terme *intro* v se réduit en $\Lambda X.v$, pourvu que l'on trouve une variable X fraîche pour v, ce qui est toujours possible.

Sous-cas $@T_2$. Du fait que l'application $@T_2\ v$ est bien typée, on déduit que v admet un type universel. D'après le lemme de classification des valeurs, il en découle que v est de la forme $\Lambda X.v'$. L'application $(@T_2)\ (\Lambda X.v')$ se réduit donc en $[X \mapsto T_2]v'$.

Sous-cas $(c_1 \to c_2)$. Du fait que l'application $(c_1 \to c_2)\ v$ est bien typée, on déduit que v admet un type flèche, donc est de la forme $\lambda x.t$. L'application $(c_1 \to c_2)\ (\lambda x.t)$ se réduit donc en $\lambda x.(c_2\ ([x \mapsto c_1\ x]t))$.

Sous-cas $\forall X.c$. Du fait que l'application $(\forall X.c)\ v$ est bien typée, on déduit que v admet un type universel, donc est de la forme $\Lambda X.v'$. (Modulo renommages, on peut faire en sorte que la variable liée dans $\forall X.c$ et celle liée dans $\Lambda X.v'$ soient identiques.) L'application $(\forall X.c)\ (\Lambda X.v')$ se réduit donc en $\Lambda X.(c\ v')$.

Sous-cas *distrib*. Du fait que l'application *distrib* v est bien typée, on déduit que v admet un type de la forme $\forall X.(T_1 \to T_2)$. Ceci implique que v est de la forme $\Lambda X.v'$, où v' admet le type $T_1 \to T_2$, donc est de la forme $\lambda x.t$. L'application *distrib* $(\Lambda X.\lambda x.t)$ se réduit donc en $\lambda x.\Lambda X.([x \mapsto @X\ x]t)$. □

6 Ajout des références au système F_η

En vue d'une éventuelle introduction des références dans le langage, on décide de restreindre le polymorphisme aux *pseudo-valeurs*, données par la grammaire $w ::= x \mid \lambda x.t \mid \Lambda X.w \mid c\ w$. La règle de typage TAbs est donc remplacée par la règle suivante :

$$\frac{\Gamma \vdash w : T \qquad X \# \Gamma}{\Gamma \vdash \Lambda X.w : \forall X.T}\ \textsc{TAbs-Pseudo-Value}$$

Question 9 *La propriété de progrès est-elle préservée ? Pourquoi ?* ◇

Solution. Le théorème de progrès reste valide, car les termes bien typés sont moins nombreux une fois la règle TAbs remplacée par la règle TAbs-Pseudo-Value. □

Question 10 *La propriété d'auto-réduction n'est pas préservée. Donner tous les points où la preuve effectuée lors de la question 7 devient invalide.* ◇

Solution. En ce qui concerne le théorème d'auto-réduction, il faut d'une part vérifier que les lemmes auxiliaires que nous avons utilisés restent valides, d'autre part vérifier la preuve du théorème.

En ce qui concerne les lemmes auxiliaires, on constate que l'énoncé du lemme de substitution des termes doit être modifié. En effet, la syntaxe des pseudo-valeurs n'est pas stable par substitution arbitraire : en général, $[x \mapsto t]w$ n'est pas une pseudo-valeur. Ceci provoque l'échec de la preuve du lemme de substitution des termes, dans le cas de la règle TAbs-Pseudo-Value. La syntaxe des pseudo-valeurs est stable par substitution de pseudo-valeurs : $[x \mapsto w_1]w_2$ est une pseudo-valeur. Par conséquent, le lemme de substitution des termes doit être restreint et devient un lemme de substitution des pseudo-valeurs :

$\Gamma; x : T_1 \vdash t : T_2$ et $\Gamma \vdash w : T_1$ impliquent $\Gamma \vdash [x \mapsto w]t : T_2$.

Heureusement, cet énoncé restreint est suffisant ; on le vérifie aux trois points où le lemme de substitution des termes était utilisé.

On relit ensuite la preuve effectuée en réponse à la question 7. Partout où l'on *déconstruisait* une dérivation de typage pour un terme de la forme $\Lambda X.t$, on peut maintenant supposer que t est une pseudo-valeur. Il faut alors vérifier que les dérivations de typage que l'on *construisait* restent valides, ce qui exige que la construction Λ ne soit appliquée qu'à des pseudo-valeurs. Une inspection des différents cas montre que tous les cas de la preuve restent valides, *sauf le cas de la coercion distrib*. Ce dernier devient incorrect, car on tente d'y construire une Λ-abstraction de la forme $\Lambda X.([x \mapsto @X\ x]t)$, où rien ne garantit que t est une pseudo-valeur. La démonstration du théorème est donc cassée, et on peut facilement construire un exemple qui montre que le théorème est effectivement faux. □

Question 11 *Quelle restriction supplémentaire pourrait-on apporter au système pour restaurer la propriété d'auto-réduction ? (On ne demande pas de refaire la preuve de la propriété d'auto-réduction une fois cette restriction effectuée.)* ◇

Solution. Pour restaurer la propriété d'auto-réduction, l'idée la plus simple consiste à supprimer la coercion *distrib*. On la retire donc de la syntaxe des coercions, et on retire les règles de réduction et de typage associées. La règle de réduction problématique disparaît ; les cas restants de la preuve d'auto-réduction sont valides. □

Question 12 *On considère le système F_η, dans lequel on remplace* TAbs *par* TAbs-Pseudo-Value*, et auquel on ajoute les trois opérations primitives pour allouer, lire et modifier une référence, dotées de leurs types standard. (On pourra également ajouter des constantes entières et booléennes, ainsi que les opérations primitives associées.) Démontrer que ce système n'est pas sûr, en exhibant un programme bien typé qui plante.* ◇

Solution. L'idée est de montrer que la coercion *distrib* est réellement dangereuse lorsque le langage est doté de références. Pour cela, on applique tout simplement *distrib* à l'opération « ref » d'allocation d'une nouvelle référence. Plus précisément, on constate que la coercion c définie comme suit :

$$\mathit{intro}; \forall X.@(X \to X); \mathit{distrib}$$

est témoin de la relation :

$$\forall X.(X \to \mathit{ref}\, X) \leq (\forall X.X \to X) \to (\forall X.\mathit{ref}\,(X \to X))$$

Il en découle que ce terme, qui alloue une nouvelle référence et y stocke la fonction identité :

$$(c\ \mathrm{ref})\ (\Lambda X.\lambda x.x)$$

admet le type :

$$\forall X.\mathit{ref}\,(X \to X)$$

Nous avons donc créé une référence polymorphe. À partir d'ici, il ne reste plus qu'à procéder comme en cours : par exemple, stocker dans cette référence au type $\mathit{int} \to \mathit{int}$ la fonction successeur, puis lire la référence au type $\mathit{bool} \to \mathit{bool}$, et appliquer la fonction ainsi obtenue à un booléen. Le terme ainsi construit plante, puisqu'il se réduit (en plusieurs étapes) vers $\mathrm{true} + 1$:

$$\begin{array}{l}\mathsf{let}\ r = (c\ \mathrm{ref})\ (\Lambda X.\lambda x.x)\ \mathsf{in}\\ @(\mathit{int} \to \mathit{int})\ r := \lambda x.x + 1;\\ !(@(\mathit{bool} \to \mathit{bool})\ r)\ \mathrm{true}\end{array}$$

En résumé, la coercion *distrib* permet de contourner la restriction du polymorphisme aux (pseudo-)valeurs. C'est pourquoi il est nécessaire de la supprimer si on souhaite introduire des références dans le langage. □

7 Une variante de la règle de distribution

On se pose la question de savoir si l'on pourrait simplifier la règle Distribution. On considère donc les deux règles suivantes :

$$\textsc{Distribution-Gauche}\quad \mathit{distrib}_g : \forall X.(T_1 \to T_2) \leq (\forall X.T_1) \to T_2$$

$$\textsc{Distribution-Droite}\quad \frac{X \# T_1}{\mathit{distrib}_d : \forall X.(T_1 \to T_2) \leq T_1 \to (\forall X.T_2)}$$

On remarque d'abord que la première de ces deux règles n'a guère d'intérêt, car elle n'apporte aucune expressivité nouvelle :

Question 13 *Démontrer que, dans le système F_η privé de la règle* Distribution*, la relation $\forall X.(T_1 \to T_2) \leq (\forall X.T_1) \to T_2$ est satisfaite. Exhiber un témoin de cette relation.* ◇

Solution. La dérivation est la suivante :

$$\dfrac{\dfrac{\textsc{$\forall$-Elimination}\quad @X : \forall X.(T_1 \to T_2) \leq T_1 \to T_2 \qquad \dfrac{\textsc{$\forall$-Elimination}\quad @X : \forall X.T_1 \leq T_1 \qquad \textsc{Réflexivité}\quad \mathit{id} : T_2 \leq T_2}{@X \to \mathit{id} : T_1 \to T_2 \leq (\forall X.T_1) \to T_2}\ \textsc{$\to$-Congruence}}{@X; (@X \to \mathit{id}) : \forall X.(T_1 \to T_2) \leq (\forall X.T_1) \to T_2}\ \textsc{Transitivité}$$

Cette dérivation démontre que $\mathit{distrib}_g$ peut être considérée comme un sucre syntaxique pour la coercion $@X; (@X \to \mathit{id})$. □

On vérifie ensuite que les règles Distribution-Droite et Distribution sont en fait équivalentes :

Question 14 *Démontrer que, dans le système F_η privé de la règle* Distribution *mais doté de la règle* Distribution-Droite*, la relation $\forall X.(T_1 \to T_2) \leq (\forall X.T_1) \to (\forall X.T_2)$ est satisfaite. Exhiber un témoin de cette relation. Réciproquement, démontrer que, dans le système F_η, la relation $\forall X.(T_1 \to T_2) \leq T_1 \to (\forall X.T_2)$ est satisfaite lorsque $X \# T_1$. Exhiber un témoin de cette relation.* ◇

Solution. Plaçons-nous dans système F_η privé de la règle Distribution mais doté de la règle Distribution-Droite. À partir d'un jugement démontré plus haut, nous déduisons :

$$\frac{@X \to id : T_1 \to T_2 \leq (\forall X.T_1) \to T_2}{\forall X.(@X \to id) : \forall X.(T_1 \to T_2) \leq \forall X.((\forall X.T_1) \to T_2)} \ \forall\text{-Congruence}$$

Par ailleurs, la règle Distribution-Droite permet de dériver :

$$\frac{X \# \forall X.T_1}{distrib_d : \forall X.((\forall X.T_1) \to T_2) \leq (\forall X.T_1) \to (\forall X.T_2)} \ \text{Distribution-Droite}$$

La règle Transitivité permet alors de conclure (la dérivation est omise) :

$$(\forall X.(@X \to id)); distrib_d : \forall X.(T_1 \to T_2) \leq (\forall X.T_1) \to (\forall X.T_2)$$

Réciproquement, plaçons-nous à présent dans le système F_η. Supposons $X \# T_1$. Nous pouvons dériver :

$$\frac{\dfrac{\dfrac{X \# T_1}{intro : T \leq \forall X.T_1}\ \forall\text{-Introduction} \qquad id : \forall X.T_2 \leq \forall X.T_2\ \text{Réflexivité}}{intro \to id : (\forall X.T_1) \to (\forall X.T_2) \leq T_1 \to (\forall X.T_2)}\ \to\text{-Congruence}$$

Les règles Distribution et Transitivité permettent alors de conclure (la dérivation est omise) :

$$distrib; (intro \to id) : \forall X.(T_1 \to T_2) \leq T_1 \to (\forall X.T_2)$$ □

Références

[1] John C. Mitchell. Polymorphic type inference and containment. *Information and Computation*, 76(2–3) :211–249, 1988.