Programmation fonctionnelle et systèmes de types (MPRI 2-4-2)
Examen 2007–2008

Xavier Leroy et François Pottier

12 février 2008

Ce problème étudie diverses propriétés de la transformation CPS, notamment la préservation de la sémantique dynamique et la préservation du typage. On signale que les questions sont souvent indépendantes les unes des autres, et que les questions les plus lointaines ne sont pas nécessairement les plus difficiles ou les plus coûteuses en temps.

Le langage considéré est le λ-calcul pur, dont on rappelle la syntaxe :

$$\begin{array}{rrcl} \text{Termes :} & a, b & ::= & x \mid \lambda x.a \mid a\ b \\ \text{Valeurs :} & v & ::= & x \mid \lambda x.a \end{array}$$

Notons que les variables x sont considérées comme faisant partie des valeurs v.

La sémantique dynamique de ce langage est donnée sous deux formes équivalentes. La sémantique par réduction se présente comme la relation $a \to a'$, «le terme a se réduit en le terme a'», définie par :

$$(\lambda x.a)\ v \to a[x \leftarrow v] \qquad \frac{a \to a'}{a\ b \to a'\ b} \qquad \frac{b \to b'}{a\ b \to a\ b'}$$

La sémantique naturelle se présente comme la relation $a \Rightarrow v$, «le terme a s'évalue en la valeur v», définie par :

$$v \Rightarrow v \qquad \frac{a \Rightarrow \lambda x.a' \qquad b \Rightarrow v' \qquad a'[x \leftarrow v'] \Rightarrow v}{a\ b \Rightarrow v}$$

Partie I. Transformation CPS et préservation de la sémantique

On considère une transformation CPS, dont la définition est donnée par les équations ci-dessous. La transformation est constituée de deux fonctions définies de façon mutuellement récursive, à savoir la traduction des valeurs $(\!|\cdot|\!)$ et la traduction des termes $[\![\cdot]\!]$.

$$\begin{array}{rcll} (\!|x|\!) & = & x & \\ (\!|\lambda x.a|\!) & = & \lambda x.[\![a]\!] & \\ [\![v]\!] & = & \lambda k.k\,(\!|v|\!) & (k \# v) \\ [\![a_1\,a_2]\!] & = & \lambda k.[\![a_1]\!]\,(\lambda x_1.[\![a_2]\!]\,(\lambda x_2.x_1\,x_2\,k)) & (k \# a_1, a_2)(x_1 \# k, a_2)(x_2 \# k, x_1) \end{array}$$

On admettra sans démonstration le résultat de commutation ci-dessous entre substitution et transformation CPS :

$$[\![a[x \leftarrow v]]\!] = [\![a]\!]\ [x \leftarrow (\!|v|\!)]$$

Le but de cette partie est de donner une preuve élémentaire (n'utilisant pas de monades) du fait que cette transformation CPS préserve la sémantique des programmes.

Question 1 *Soit a un terme, v une valeur et $K = \lambda k \ldots$ une valeur représentant une continuation. Supposons que $a \Rightarrow v$. Montrer que $[\![a]\!]\ K \xrightarrow{*} K\ (\!|v|\!)$.* ◇

Question 2 *Soit a un programme qui s'évalue en la valeur v : $a \Rightarrow v$. Comment s'évalue le programme après transformation CPS, à savoir le terme $[\![a]\!]\ (\lambda x.x)$?* ◇

Partie II. Transformation CPS et préservation des types

On suppose les langages source et cible de la transformation simplement typés. La grammaire des types est :

$$\tau ::= \alpha \mid \tau \to \tau$$

Question 3 *On fixe une variable de types o pour représenter le type des réponses, c'est-à-dire le codomaine des continuations. Définir deux fonctions de traduction des types, que nous noterons également $(\!|\cdot|\!)$ et $[\![\cdot]\!]$, de façon à ce que les énoncés suivants soient satisfaits :*

1. *si $\Gamma \vdash v : \tau$, alors $(\!|\Gamma|\!) \vdash (\!|v|\!) : (\!|\tau|\!)$;*
2. *si $\Gamma \vdash a : \tau$, alors $(\!|\Gamma|\!) \vdash [\![a]\!] : [\![\tau]\!]$.*

Démontrer alors ces deux énoncés. ◇

Question 4 *Quel rôle a joué la définition de la traduction des variables de types dans la démonstration précédente ? Pourquoi ?* ◇

On se place maintenant dans Système F explicitement typé, avec restriction du polymorphisme aux valeurs. La grammaire des valeurs, termes et types est étendue comme suit :

$$\begin{aligned} v &::= \ldots \mid \Lambda\alpha.v \\ a &::= \ldots \mid a\,\tau \\ \tau &::= \ldots \mid \forall\alpha.\tau \end{aligned}$$

Question 5 *Étendre les traductions des valeurs, des termes, et des types. Étendre la démonstration du fait que la traduction préserve le typage. Quelle propriété de la fonction de traduction des types exploite-t-on ? Quel rôle joue maintenant la définition de la traduction des variables de types ?* ◇

Question 6 *Peut-on tirer parti de l'expressivité de Système F pour éviter de fixer un type o des réponses ? Comment modifier alors la définition de la traduction des types ? (On ne demande pas d'expliquer comment étendre la traduction des termes ni la démonstration de la propriété de préservation du typage.)* ◇

Partie III. Exceptions et transformation CPS «à deux canons»

Dans cette partie, on étend le langage d'entrée de la transformation CPS avec les constructions $\mathsf{raise}(a)$ pour lever une exception et $\mathsf{try}\ a\ \mathsf{with}\ x \to b$ pour récupérer les exceptions levées pendant l'évaluation de a.

Termes :	a, b	$::=$	$v \mid a\ b \mid \mathsf{raise}(a) \mid \mathsf{try}\ a\ \mathsf{with}\ x \to b$
Valeurs :	v	$::=$	$x \mid \lambda x.a$
Résultats d'évaluations	r	$::=$	$v \mid \mathsf{raise}(v)$

La sémantique naturelle de ce langage est donnée par la relation $a \Rightarrow r$, où r est ou bien une valeur v («le terme a s'évalue en la valeur v sans lever d'exception non rattrapée») ou bien une levée d'exception $\mathsf{raise}(v)$ («l'évaluation du terme a se termine abruptement en levant l'exception v»).

$$v \Rightarrow v \qquad \frac{a \Rightarrow \lambda x.a' \qquad b \Rightarrow v \qquad a'[x \leftarrow v] \Rightarrow r}{a\ b \Rightarrow r} \qquad \frac{a \Rightarrow \mathsf{raise}(v)}{a\ b \Rightarrow \mathsf{raise}(v)} \qquad \frac{a \Rightarrow v_1 \qquad b \Rightarrow \mathsf{raise}(v_2)}{a\ b \Rightarrow \mathsf{raise}(v_2)}$$

$$\frac{a \Rightarrow v}{\mathsf{raise}(a) \Rightarrow \mathsf{raise}(v)} \qquad \frac{a \Rightarrow \mathsf{raise}(v)}{\mathsf{raise}(a) \Rightarrow \mathsf{raise}(v)} \qquad \frac{a \Rightarrow v}{(\mathsf{try}\ a\ \mathsf{with}\ x \to b) \Rightarrow v} \qquad \frac{a \Rightarrow \mathsf{raise}(v) \qquad b[x \leftarrow v] \Rightarrow r}{(\mathsf{try}\ a\ \mathsf{with}\ x \to b) \Rightarrow r}$$

On définit une traduction CPS de ce langage étendu avec des exceptions vers le λ-calcul sans exceptions, en suivant l'approche connue sous le nom de «CPS à deux canons». La traduction CPS $[\![a]\!]$ d'un terme a est une fonction $\lambda k_1 k_2 \ldots$ qui prend en arguments non pas une continuation k comme dans la partie I, mais deux continuations k_1 et k_2.

– La première continuation k_1 est la continuation «normale». Si l'évaluation de a termine sur la valeur v sans lever d'exception, la traduction $[\![a]\!]\ k_1\ k_2$ applique la continuation k_1 à la valeur $(\!|v|\!)$ correspondant à v.
– La seconde continuation k_2 est la continuation d'exception. Si l'évaluation de a termine en levant une exception de valeur v, la traduction $[\![a]\!]\ k_1\ k_2$ applique la continuation k_2 à la valeur $(\!|v|\!)$.

Voici la définition de la traduction CPS «à deux canons» pour les valeurs et les applications :

$$\begin{array}{rcl} (\!|x|\!) & = & x \\ (\!|\lambda x.a|\!) & = & \lambda x.[\![a]\!] \\ \\ [\![v]\!] & = & \lambda k_1 k_2.k_1\,(\!|v|\!) \\ [\![a_1\,a_2]\!] & = & \lambda k_1 k_2.[\![a_1]\!]\,(\lambda x_1.[\![a_2]\!]\,(\lambda x_2.x_1\,x_2\,k_1\,k_2)\,k_2)\,k_2 \end{array}$$

Question 7 *Définir la traduction CPS «à deux canons» pour les constructions* `raise` *et* `try...with` *:*

$$\begin{array}{rcl} [\![\mathsf{raise}(a)]\!] & = & \lambda k_1 k_2.\ ??? \\ [\![\mathsf{try}\ a\ \mathsf{with}\ x \to b]\!] & = & \lambda k_1 k_2.\ ??? \end{array}$$

On pourra raisonner par cas selon que a termine normalement ou bien lève une exception. ◇

Question 8 *Énoncer (sans démonstration pour l'instant) une propriété de préservation sémantique analogue à celle de la question 1. Plus précisément : si $a \Rightarrow r$ et si K_1 et K_2 sont des valeurs, en quel terme se réduit $[\![a]\!]\ K_1\ K_2$?* ◇

On admettra à nouveau sans démonstration le résultat de commutation ci-dessous entre substitution et transformation CPS :

$$[\![a[x \leftarrow v]]\!] = [\![a]\!]\ [x \leftarrow (\!|v|\!)]$$

Question 9 *Démontrer l'énoncé de préservation sémantique de la question précédente dans les cas suivants : (1) a est une valeur ; (2) $a = \mathsf{try}\ a_1\ \mathsf{with}\ x \to a_2$.* ◇

Partie IV. Transformation CPS «à deux canons» et préservation du typage

On définit un système de types simples pour le λ-calcul enrichi avec des exceptions de la partie III. Les types sont les suivants :

$$\begin{array}{rcl} \tau & ::= & \alpha \mid \tau \to \phi \\ \phi & ::= & \tau + \tau \end{array}$$

Les jugements de typage sont de la forme suivante :

$$\begin{array}{ll} \Gamma \vdash v : \tau & \text{(valeurs)} \\ \Gamma \vdash a : \phi & \text{(termes)} \end{array}$$

Un ¡¡type de résultat¿¿ ϕ est de la forme $\tau_1 + \tau_2$, où τ_1 est le type de la valeur produite, si le terme considéré termine normalement, et τ_2 est le type de l'exception produite, si le terme considéré lance une exception. L'environnement de typage Γ associe aux variables des types ordinaires τ.

Question 10 *Donner les six règles de typage correspondant aux deux formes de valeurs et aux quatre formes d'expressions. (On ne demande pas de démonstration.)* ◇

Question 11 *Le langage source de la traduction CPS à deux canons est le λ-calcul simplement typé décrit ci-dessus, tandis que le langage cible reste le λ-calcul simplement typé standard. Définir les fonctions de traduction des types. (On ne demande pas de démonstration.)* ◇

Références

[1] Josh Berdine, Peter W. O'Hearn, Uday S. Reddy, and Hayo Thielecke. Linear continuation-passing. *Higher-Order and Symbolic Computation*, 15(2–3) :181–208, 2002.

[2] Andrew Kennedy. Compiling with continuations, continued. In *ACM International Conference on Functional Programming (ICFP)*, pages 177–190, September 2007.

[3] Hayo Thielecke. Comparing control constructs by double-barrelled CPS. *Higher-Order and Symbolic Computation*, 15(2–3) :141–160, 2002.

[4] Hayo Thielecke. From control effects to typed continuation passing. In *ACM Symposium on Principles of Programming Languages (POPL)*, pages 139–149, January 2003.