Typage des objets

Second examen partiel, MPRI 2-4

2021/01/06 — durée: 1h30

1 Calcul d'objets

On s'intéresse au ς-calcul qui permet de modéliser les constructions élémentaires des langages orientés objets. Les expressions du ς-calcul sont définies par la grammaire suivante :

$$a \quad ::= \quad x \quad | \quad \{\ell_i = \varsigma(x)\, a_i^{i \in I}\} \quad | \quad a.\ell \quad | \quad a.\ell \Leftarrow \varsigma(x)\, a$$

Outre les variables x, on a donc trois constructions syntaxiques, qui représentent respectivement la construction d'un objet, donné par la liste de ses méthodes (I est un ensemble d'indices, possiblement vide, typiquement un ensemble d'entiers de 1 à n) ; l'appel de la méthode ℓ de l'objet a ; et la redéfinition de la méthode ℓ de l'objet a. La construction $\varsigma(x)\, a$ lie la variable x dans l'expression a (et les expressions sont donc considérées égales par renommage des variables liées). Informellement, la variable x représente l'objet lui-même ("self"). Les méthodes n'attendent aucun argument hormis l'objet lui-même.

La sémantique du ς-calcul est donnée par les règles de réduction suivantes, où j est dans I,

$$\begin{array}{rcll}
\{\ell_i = \varsigma(x)\, a_i^{i \in I}\}.\ell_j & \rightsquigarrow & a_j[x \leftarrow \{\ell_i = \varsigma(x)\, a_i^{i \in I}\}] & \textsc{R-Send} \\
\{\ell_i = \varsigma(x)\, a_i\}.\ell_j \Leftarrow \varsigma(x)\, a' & \rightsquigarrow & \{\ell_i = \varsigma(x)\, a_i^{i \in I \setminus \{j\}}, \ell_j = \varsigma(x)\, a'\} & \textsc{R-Over} \\
E[a] \rightsquigarrow E[a'] & \text{if} & a \rightsquigarrow a' & \textsc{R-Context}
\end{array}$$

Les valeurs v et les contextes de réduction sont définis par :

$$v ::= \{\ell_i = \varsigma(x)\, a_i^{i \in I}\}$$
$$E ::= [\,].\ell \mid [\,].\ell \Leftarrow \varsigma(x)\, a$$

Pour alléger les notations, on note a au lieu de $\varsigma(x)\, a$ si x n'est pas libre dans a. On définit les expressions suivantes :

$$\begin{array}{rcl}
a_0 & \triangleq & \{\ell = \varsigma(x)\, x.\ell\}.\ell \\
a_1 & \triangleq & \left\{\ell_0 = a_0; \ell_1 = \varsigma(x)\, \left(x.\ell_0 \Leftarrow \{\}\right).\ell_0\right\}.\ell_1
\end{array}$$

Question 1

a) Donner les séquences de réduction pour les expressions a_0 et a_1.

b) Donner un exemple de programme bloqué. $\square$

On munit le ς-calcul d'un système de types simples. Les types et environnements de typage sont les suivants :

$$\begin{array}{rcll} \tau & ::= & \{\ell_i = \tau_i^{i \in I}\} & \text{types} \\ \Gamma & ::= & \emptyset \mid \Gamma, x : \tau & \text{environnements} \end{array}$$

Notez l'absence de variables de types. Nous ne manipulerons donc que des types clos.

On définit le jugement $\Gamma \vdash a : \tau$ comme la plus petite relation satisfaisant les règles suivantes :

$$\textsc{Object}\ \frac{\tau = \{\ell_i : \tau_i{}^{i \in I}\} \qquad \forall i \in I \quad \Gamma, x : \tau \vdash a_i : \tau_i}{\Gamma \vdash \{\ell_i = \varsigma(x)\, a_i{}^{i \in I}\} : \tau} \qquad \textsc{Send}\ \frac{\Gamma \vdash a : \{\ell_i : \tau_i{}^{i \in I}\} \qquad j \in I}{\Gamma \vdash a.\ell_j : \tau_j}$$

$$\textsc{Over}\ \frac{\tau = \{\ell_i : \tau_i{}^{i \in I}\} \qquad \Gamma \vdash a : \tau \qquad \Gamma, x : \tau \vdash a' : \tau_j \qquad j \in I}{\Gamma \vdash a.\ell_j \Leftarrow \varsigma(x)\, a' : \tau} \qquad \textsc{Var}\ \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}$$

Question 2

a) Donner un exemple d'expression mal typée (sans justification).

b) Donner les types et les dérivations de typage pour les expressions a_0 et a_1. (On pourra utiliser le fait que $\vdash a : \tau$ implique $\Gamma \vdash a : \tau$.) □

On admet le lemme de substitution, dont l'énoncé est le suivant : *si* $\Gamma, x : \tau', \Gamma' \vdash a : \tau$ *et* $\Gamma \vdash a' : \tau'$, *alors* $\Gamma, \Gamma' \vdash a[x \leftarrow a'] : \tau$.

Question 3

On veut montrer que la règle de réduction R-Send préserve le typage. a) Énoncez la propriété. b) Montrez-la. □

On dit qu'une traduction d'un langage L_1 dans un langage L_2 est une bisimulation si on peut fournir une fonction de traduction totale $[\![\cdot]\!]$ préservant la sémantique, c'est-à-dire :

- — les valeurs de L_1 se traduisent en des valeurs de L_2
- — Si $a \rightsquigarrow a'$ dans L_1 alors $[\![a]\!] \rightsquigarrow^+ [\![a']\!]$ dans L_2.
- — Si $[\![a]\!] \rightsquigarrow e$ dans L_2 alors $a \rightsquigarrow a'$ dans L_1 et $e \rightsquigarrow^* [\![a']\!]$.

Question 4

Peut-on donner une traduction du ς-calcul simplement typé dans le λ-calcul simplement typé avec juste une constante unité () de type unit qui soit une bisimulation ? (On justifiera brièvement la réponse.) □

Inversement, on définit la traduction suivante du λ-calcul dans le ς-calcul, où val et arg sont deux étiquettes arbitraires distinctes fixées. On note e les expressions du λ-calcul implicitement et simplement typé, étendu avec une seule constante unité () de type unit.

$$[\![()]\!] \triangleq \{\} \qquad [\![x]\!] \triangleq x.\mathsf{arg} \qquad [\![\lambda x.\, e]\!] \triangleq \{\mathsf{arg} = \bot, \mathsf{val} = \varsigma(x)\, [\![e]\!]\}$$

$$[\![e_1\ e_2]\!] \triangleq ([\![e_1]\!].\mathsf{arg} \Leftarrow [\![e_2]\!]).\mathsf{val}$$

où $\perp$ est un terme du ς-calcul qui admet tous les types.

Question 5
Quel terme peut-on prendre pour $\perp$? $\square$

Question 6
On note e_1 le terme $(\lambda x.\, x)\ ()$ du λ-calcul.
a) Donnez la réduction, pas à pas, de $[\![e_1]\!]$ jusqu'à obtenir une valeur v du ς-calcul.
b) Combien la réduction contient-elle de pas élémentaires ?
c) Que suggère cet exemple en lien avec la définition d'une bisimulation ? $\square$

Question 7
Répondre aux trois mêmes questions que l'exercice précédent pour le terme e_2 égal à $(\lambda x.\, \lambda z.\, x)\ ()$. $\square$

Question 8
Quelle stratégie d'évaluation du λ-calcul cette traduction vers le ς-calcul implémente-t-il ? (On justifiera brièvement la réponse.) $\square$

On souhaite à présent montrer que la trauction présentée ci-dessus préserve le typage. On notera σ les types du λ-calcul simplement typé, définis par :

$$\sigma \ ::= \ \mathsf{unit} \mid \sigma \to \sigma$$

Question 9
Donner la traduction des types du λ-calcul vers les types du ς-calcul. $\square$

On note Δ les environnements de typage du λ-calcul et on traitera la constante $()$ de façon primitive avec l'axiome $\textsc{Unit}$: $\Delta \vdash () : \mathsf{unit}$. Les autres règles de typage sont celles du λ-calcul simplement typé. Nous voulons montrer la préservation du typage : Si $\vdash e : \sigma$ alors $\vdash [\![e]\!] : [\![\sigma]\!]$. Pour cela, il nous faut renforcer la propriété avec des termes ouverts. Cependant la traduction $[\![\Delta]\!]$ n'est pas uniquement un environnement de typage, mais un ensemble d'environnements de typages (pour une raison à découvrir). On pourra alors montrer

Si $\Delta \vdash e : \sigma$ alors $\Gamma \vdash [\![e]\!] : [\![\sigma]\!]$ pour tout Γ dans $[\![\Delta]\!]$.

Question 10
a) Donner la définition de $[\![\Delta]\!]$ qui permette de montrer la propriété ci-dessus.
b) Montrer la propriété ci-dessus. (On donnera clairement le schéma de preuve, mais on pourra ne traiter que les cas des variables et des abstractions.) $\square$

Question 11
a) Pourquoi, bien que bien typé, cette traduction n'est-elle pas entièrement satisfaisante ?
b) Que pourrait-on attendre du typage ? $\square$