Examen du cours “Typage et programmation”

D.E.A. “Programmation”

9 décembre 1999

Exercice

Le but de cet exercice est d'étendre le langage mini-ML pour effectuer des vérifications de types dynamiques, c'est-à-dire pendant l'exécution.

Suivant l'approche proposée en 1986 par Luca Cardelli, nous ajoutons au langage des opérateurs manipulant des *valeurs avec types dynamiques*, ou *dynamiques* en abrégé, c'est-à-dire des paires (v, τ) d'une valeur v et d'une représentation du type τ de v. Pour calculer sur ces dynamiques, on se donne les trois familles d'opérateurs suivantes :

- $\texttt{dyn}_\tau(a)$ pour construire un dynamique dont la partie valeur est la valeur de a et la partie type est τ ;
- $\texttt{hastype}_\tau(a)$, qui renvoie le booléen `true` si a est un dynamique dont la partie type est égale à τ, et `false` sinon ;
- $\texttt{coerce}_\tau(a)$, qui vérifie que a est un dynamique dont la partie type est τ, renvoie sa partie valeur si c'est le cas, et ne se réduit pas sinon.

Par exemple, voici une fonction qui essaye d'imprimer son argument (un dynamique) par discrimination sur son type :

```
fun d →
  if hastype_string(d) then print_string(coerce_string(d))
  else if hastype_int(d) then print_int(coerce_int(d))
  else if hastype_bool(d) then
                print_string (if coerce_bool(d) then "true" else "false")
  else print_string "???"
```

Les règles de réduction et les schémas de types pour ces opérateurs sont les suivants :

$$\texttt{hastype}_\tau(\texttt{dyn}_{\tau'}(v)) \xrightarrow{\varepsilon} \texttt{true} \quad \text{si } \tau = \tau' \tag{1}$$
$$\texttt{hastype}_\tau(\texttt{dyn}_{\tau'}(v)) \xrightarrow{\varepsilon} \texttt{false} \quad \text{si } \tau \neq \tau' \tag{2}$$
$$\texttt{coerce}_\tau(\texttt{dyn}_{\tau'}(v)) \xrightarrow{\varepsilon} v \quad \text{si } \tau = \tau' \tag{3}$$

$$\begin{array}{rcll}
\texttt{dyn}_\tau & : & \tau \to \texttt{dyn} & \text{si } \tau \text{ est un type sans variables de types} \\
\texttt{hastype}_\tau & : & \texttt{dyn} \to \texttt{bool} & \\
\texttt{coerce}_\tau & : & \texttt{dyn} \to \tau & \text{si } \tau \text{ est un type sans variables de types}
\end{array}$$

Dans les types ci-dessus, `dyn` est un nouveau type de base (comme `int` ou `bool`) qui représente le type statique de tous les dynamiques.

Enfin, on étend la classe syntaxique des valeurs pour considérer que $\mathtt{dyn}_\tau(v)$ est une valeur (un dynamique entièrement évalué) si v est une valeur :

Valeurs : $v \ ::= c \mid op \mid \mathtt{fun}\ x \to a \mid (v_1, v_2) \mid \mathtt{dyn}_\tau(v)$

Question 1. Montrer que l'hypothèse (H1) de la preuve de sûreté du chapitre 2 du cours est vérifiée pour les règles de réduction (1), (2) et (3). C'est-à-dire, montrer que si $E \vdash a : \tau$ et $a \xrightarrow{\varepsilon} a'$ par l'une des règles (1), (2) ou (3), alors $E \vdash a' : \tau$.

Question 2. Montrer l'hypothèse (H2) pour l'opérateur $\mathtt{hastype}_\tau$. C'est-à-dire, montrer que si $\emptyset \vdash \mathtt{hastype}_\tau(v) : \tau'$, alors le terme $\mathtt{hastype}_\tau(v)$ peut se réduire.

Question 3. Est-ce que l'hypothèse (H2) est vérifiée pour l'opérateur $\mathtt{coerce}_\tau$? Si oui, la démontrer. Si non, donner un contre-exemple et proposer un moyen pour corriger ce problème : ajout d'une ou plusieurs règles de réduction supplémentaires, ou bien remplacement de $\mathtt{coerce}_\tau$ par un opérateur ayant la même puissance d'expression, mais vérifiant (H2).

Question 4. Supposons que l'on lève la restriction que le type τ dans $\mathtt{dyn}_\tau$ et $\mathtt{coerce}_\tau$ ne doit pas contenir de variables de types. Par exemple, on autoriserait $\tau = \alpha \to \alpha$, avec les schémas de types suivants pour $\mathtt{dyn}_\tau$ et $\mathtt{coerce}_\tau$:

$$\begin{array}{rcl} \mathtt{dyn}_{\alpha\to\alpha} & : & \forall\alpha.\ (\alpha \to \alpha) \to \mathtt{dyn} \\ \mathtt{coerce}_{\alpha\to\alpha} & : & \forall\alpha.\ \mathtt{dyn} \to (\alpha \to \alpha) \end{array}$$

Montrer par un exemple que cela "casse" la sûreté du typage (l'hypothèse (H1) ne serait plus vérifiée).

Problème

Le but de ce problème est de montrer que si l'on peut donner un type "suffisamment polymorphe" à une fonction, alors on peut deviner à partir du type quel est le comportement de la fonction. Par exemple, si $\emptyset \vdash a : \alpha \to \alpha$, alors a est forcément ou bien la fonction identité, ou bien une fonction qui ne termine jamais. De tels résultats s'appellent des propriétés de paramétricité.

Dans tout ce problème, on se place dans le langage mini-ML du chapitre 2, avec une seule sorte de constantes c : les constantes entières, et un seul type de base T : le type `int` des entiers. On se donne comme opérateurs *op* les projections `fst` et `snd`, les opérations arithmétiques usuelles sur les entiers, et le point fixe `fix`.

Préambule

Question 1. On considère l'expression $\omega = \mathtt{fix}(\mathtt{fun}\ x \to x)$. Montrer que $\emptyset \vdash \omega : \alpha$. Montrer que ω se réduit en ω, et que donc son évaluation ne termine jamais.

Question 2. Soit a une expression telle que $\emptyset \vdash a : \alpha$. Montrer que a diverge, c'est-à-dire que l'évaluation de a conduit forcément à une suite infinie de réductions $a \to a_1 \to a_2 \to \ldots$. (Indication : supposer que $a \xrightarrow{*} v$ où v est une valeur. Quel est le type de v ? En déduire une contradiction.)

Question 3. Donner deux exemples d'expressions a telles que $\emptyset \vdash a : \beta \to \alpha$.

Question 4. Soit a une expression telle que $\emptyset \vdash a : \beta \to \alpha$. Montrer que a est une fonction qui ne termine jamais, c'est-à-dire que $a\ b$ diverge pour tout argument b. (Indication : quel est le type de $a\ b$?)

Question 5. Donner trois exemples d'expressions a telles que $\emptyset \vdash a : \alpha \to \alpha$.

Définition de l'équivalence par relations logiques

Dans la suite de ce problème, nous allons montrer que toute expression a de type $\alpha \to \alpha$ est l'un des trois exemples trouvés à la question 5. Pour ce faire, nous allons montrer un puissant théorème sémantique appelé "lemme fondamental des relations logiques" et dû à Rick Statman (1985).

Une *interprétation des variables de types* est une fonction finie ρ qui associe à certaines variables de types α un ensemble (fini ou infini) de paires de valeurs $\rho(\alpha) = \{(v_1, v'_1); (v_2, v'_2); \ldots\}$.

On dit que deux expressions a et a' sont équivalentes en le type τ et dans l'interprétation ρ, et on note $\rho \vdash a \approx a' : \tau$, si et seulement si

- ou bien a diverge et a' diverge ;
- ou bien il existe deux valeurs v et v' telles que $a \xrightarrow{*} v$ et $a' \xrightarrow{*} v'$ et $\rho \vdash v \approx v' : \tau$ (les deux expressions s'évaluent en deux valeurs équivalentes en τ).

L'équivalence entre valeurs $\rho \vdash v \approx v' : \tau$ est définie par cas sur le type τ, comme suit :

- Si τ est le type `int` des entiers :

$$\rho \vdash n \approx n' : \mathtt{int} \text{ si et seulement si } n = n'$$

(Deux valeurs entières sont équivalentes si et seulement si elles sont égales.)

– Si τ est une variable de type α :

$$\rho \vdash v \approx v' : \alpha \text{ si et seulement si } \alpha \in \mathrm{Dom}(\rho) \text{ et } (v, v') \in \rho(\alpha)$$

(Deux valeurs d'une variable de type α sont équivalentes si et seulement si elles sont dans l'interprétation $\rho(\alpha)$ de cette variable.)

– Si τ est un type de fonction $\tau_1 \rightarrow \tau_2$:

$$\rho \vdash v \approx v' : \tau_1 \rightarrow \tau_2 \text{ ssi pour toutes valeurs } w,\, w', \quad \rho \vdash w \approx w' : \tau_1 \;\Rightarrow\; \rho \vdash v\ w \approx v'\ w' : \tau_2$$

(Deux fonctions ou opérateurs sont équivalents si et seulement si ils envoient des arguments équivalents sur des résultats équivalents.)

– Si τ est un type produit $\tau_1 \times \tau_2$:

$$\rho \vdash (v_1, v_2) \approx (v_1', v_2') : \tau_1 \times \tau_2 \text{ si et seulement si } \rho \vdash v_1 \approx v_1' : \tau_1 \text{ et } \rho \vdash v_2 \approx v_2' : \tau_2$$

(Deux paires sont équivalentes si et seulement si leurs composantes sont deux à deux équivalentes.)

Enfin, on étend l'équivalence entre expressions aux schémas de types de la manière suivante :

$$\rho \vdash a \approx a' : \forall \alpha_1, \ldots, \alpha_n.\ \tau$$

si et seulement si, pour des ensembles $R_1, \ldots, R_n$ arbitraires de paires de valeurs, on a

$$\rho + \{\alpha_1 \leftarrow R_1; \ldots; \alpha_n \leftarrow R_n\} \vdash a \approx a' : \tau$$

Question 6. Est-ce que

$$\emptyset \vdash (\texttt{fun}\ x \rightarrow x) \approx (\texttt{fun}\ x \rightarrow x + 1) : \texttt{int} \rightarrow \texttt{int}\,?$$

Même question pour

$$\emptyset \vdash (\texttt{fun}\ x \rightarrow x + x) \approx (\texttt{fun}\ x \rightarrow x * 2) : \texttt{int} \rightarrow \texttt{int}.$$

(Répondre informellement ; il n'est pas nécessaire de faire des démonstrations.)

Le lemme fondamental des relations logiques

Question 7. On considère l'évaluation d'une paire (a, b). Discuter la forme des réductions possibles pour (a, b) dans les trois cas suivants :

(i) a diverge
(ii) $a \xrightarrow{*} v$ et b diverge
(iii) $a \xrightarrow{*} v$ et $b \xrightarrow{*} w$.

En déduire que si $\rho \vdash a \approx a' : \tau_1$ et $\rho \vdash b \approx b' : \tau_1$, alors $\rho \vdash (a, b) \approx (a', b') : \tau_1 \times \tau_2$.

Question 8. On considère l'évaluation d'une application $a\ b$. Discuter la forme des réductions possibles pour $a\ b$ dans les trois cas suivants :

(i) a diverge
(ii) $a \xrightarrow{*} v$ et b diverge
(iii) $a \xrightarrow{*} v$ et $b \xrightarrow{*} w$.

En déduire que si $\rho \vdash a \approx a' : \tau_1 \rightarrow \tau_2$ et $\rho \vdash b \approx b' : \tau_1$, alors $\rho \vdash a\ b \approx a'\ b' : \tau_2$.

Question 9. On note $R(\tau) = \{(v, v') \mid \rho \vdash v \approx v' : \tau\}$. On admettra le lemme de substitution suivant pour la relation d'équivalence : si

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

alors

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

Montrer que si $\rho \vdash a \approx a' : \sigma$ et si le type τ est une instance du schéma σ, alors $\rho \vdash a \approx a' : \tau$.

Question 10 (le lemme fondamental). Supposons $E \vdash a : \tau$ avec $\mathrm{Dom}(E) = \{x_1 \ldots x_n\}$. Soient $v_1, \ldots, v_n$ et $v'_1, \ldots, v'_n$ des valeurs telles que

$$\rho \vdash v_i \approx v'_i : E(x_i) \text{ pour tout } i = 1, \ldots, n$$

Montrer que

$$\rho \vdash a[x_1 \leftarrow v_1, \ldots, x_n \leftarrow v_n] \approx a[x_1 \leftarrow v'_1, \ldots, x_n \leftarrow v'_n] : \tau$$

Autrement dit, pour tout terme bien typé a, si l'on remplace ses variables libres par deux jeux de valeurs qui sont équivalentes, on obtient deux termes équivalents. (Indication : on procédera par récurrence sur la dérivation de $E \vdash a : \tau$ et par cas sur a. On détaillera les cas suivants :
- a est un identificateur x
- a est une constante n
- a est une fonction $\texttt{fun}\ x \to b$
- a est une application $b\ c$
- a est une paire (b, c).

On admettra les autres cas, et on utilisera les résultats des questions 7, 8 et 9.)

Application du lemme fondamental : résultats de paramétricité

Question 11. Soit a un terme tel que $\emptyset \vdash a : \alpha \to \alpha$. Montrer que a tombe forcément dans l'un des trois cas suivants :
- a diverge
- a s'évalue en une fonction qui ne termine jamais
- a s'évalue en la fonction identité, c'est-à-dire, la fonction f telle que $f\ v \xrightarrow{*} v$ pour tout v.

(Indication : si $a \xrightarrow{*} f$ où f est une valeur fonctionnelle, discuter suivant que $f\ 0$ diverge ou non. Si $f\ 0$ diverge, montrer que $f\ v$ diverge pour toute valeur v en interprétant α par l'ensemble vide. Si $f\ 0$ se réduit en une valeur, montrer pour toute valeur v que $f\ v \xrightarrow{*} v$ en interprétant α par l'ensemble $\{(0, v)\}$.)

Question 12. Soit a un terme tel que $\emptyset \vdash a : \alpha \times \beta \to \alpha$. Montrer que a tombe forcément dans l'un des cas suivants :
- a diverge
- a s'évalue en une fonction qui ne termine jamais
- a se comporte comme la première projection `fst`, c'est-à-dire $a\ (v, w) \xrightarrow{*} v$ pour toutes valeurs v et w.

Question 13. Soit a un terme tel que $\emptyset \vdash a : (\alpha \to \alpha) \to (\alpha \to \alpha)$. Montrer que a tombe forcément dans l'un des cas suivants :

- a diverge
- a s'évalue en une fonction qui ne termine jamais
- a s'évalue en une fonction f vérifiant la propriété suivante : il existe un entier n tel que pour toute fonction totale g (c'est-à-dire, $g\ v$ termine toujours quel que soit v) et toute valeur v, $f\ g\ v$ se réduit en le résultat de $g(g(\dots g(v)))$ (n applications de g). Autrement dit, $f\ g$ se comporte comme la composée $g \circ g \circ \cdots \circ g$ (composition itérée n fois).

(Indication : on considérera f `succ` 0, où `succ` est la fonction successeur $\texttt{fun}\ x \to x+1$, et on interprétera α par l'ensemble $\{(0, v); (1, v_1); (2, v_2); \dots\}$, où v_1 est le résultat de $g\ v$, v_2 le résultat de $g\ v_1$, etc.)