Corrigé de l'examen du cours "Typage et programmation"

D.E.A. "Programmation"

14 décembre 2000

Question 1 À chaque étape d'évaluation, on décompose l'expression sous la forme d'un contexte Γ appliqué à une expression qui peut se réduire en tête. Si $n \neq 0$, on obtient la séquence de réductions suivante:

$$\begin{array}{cl}
\texttt{try}\ (1, /(100, n))\ \texttt{with}\ x \to (0,0) & \\
\downarrow & \text{par la règle } (\delta_/) \text{ dans le contexte } \texttt{try}\ (1, [\,])\ \texttt{with}\ x \to (0,0) \\
\texttt{try}\ (1, n')\ \texttt{with}\ x \to (0,0) & \text{avec } n' = 100/n \\
\downarrow & \text{par la règle } (\text{try}_1) \text{ dans le contexte } [\,] \\
(1, n') &
\end{array}$$

C'est le résultat du calcul car $(1, n')$ est une valeur. Si $n = 0$, la séquence de réduction est:

$$\begin{array}{cl}
\texttt{try}\ (1, /(100, 0))\ \texttt{with}\ x \to (0,0) & \\
\downarrow & \text{par la règle } (\delta_{/0}) \text{ dans le contexte } \texttt{try}\ (1, [\,])\ \texttt{with}\ x \to (0,0) \\
\texttt{try}\ (1, \texttt{raise divzero})\ \texttt{with}\ x \to (0,0) & \\
\downarrow & \text{par la règle } (\text{propagation}_5) \text{ dans le contexte } \texttt{try}\ [\,]\ \texttt{with}\ x \to (0,0) \\
\texttt{try raise divzero with}\ x \to (0,0) & \\
\downarrow & \text{par la règle } (\text{try}_2) \text{ dans le contexte } [\,] \\
(0,0)\{x \leftarrow \texttt{divzero}\} = (0,0) &
\end{array}$$

Ce terme est une valeur et est le résultat de l'évaluation.

Question 2 Il faut vérifier que si $E \vdash a : \tau$ et $a \xrightarrow{\varepsilon} a'$ par l'une des règles (match$_1$) ou (match$_2$), alors $E \vdash a' : \tau$. Considérons d'abord la règle (match$_1$). On a $a = (\texttt{match V}\ v\ \texttt{with V}\ x \to a_2 \mid \texttt{E}\ y \to a_3)$ et $a' = a_2\{x \leftarrow v\}$. Vu les règles de typage, la dérivation de $E \vdash a : \tau$ est nécessairement de la forme suivante:

$$\dfrac{\dfrac{\dfrac{\tau_1 \to (\tau_1, \tau_2)\ \texttt{res} \leq TC(\texttt{V})}{E \vdash \texttt{V}\ : \tau_1 \to (\tau_1, \tau_2)\ \texttt{res}} \qquad E \vdash v : \tau_1}{E \vdash \texttt{V}\ v : (\tau_1, \tau_2)\ \texttt{res}} \qquad E + \{x : \tau_1\} \vdash a_2 : \tau \qquad E + \{y : \tau_2\} \vdash a_3 : \tau}{E \vdash \texttt{match V}\ v\ \texttt{with V}\ x \to a_2 \mid \texttt{E}\ y \to a_3 : \tau}$$

Nous avons donc des dérivations de $E \vdash v : \tau_1$ et $E + \{x : \tau_1\} \vdash a_2 : \tau$. Appliquant le lemme de substitution vu dans le cours, il vient que $E \vdash a_2\{x \leftarrow v\} : \tau$. C'est le résultat attendu: $E \vdash a' : \tau$.

La preuve pour la règle (match$_2$) est identique en considérant a_3 au lieu de a_2 et E au lieu de V.

Question 3 Remarquons tout d'abord que les seules valeurs de type (τ_1, τ_2) `res` sont de la forme `V` v_1 ou `E` v_2, avec dans le premier cas v_1 de type τ_1 et dans le second cas v_2 de type τ_2. En effet, les autres valeurs (constantes, opérateurs, fonctions, paires) ont des types qui ne sont pas de la forme (τ_1, τ_2) `res` (respectivement, type constant, type flèche, type flèche et type produit).

Maintenant, supposons a bien typée dans l'environnement vide. Vu les règles de typage, nous avons nécessairement une dérivation de typage de la forme suivante:

$$\frac{\emptyset \vdash v : (\tau_1, \tau_2)\ \texttt{res} \qquad \{x : \tau_1\} \vdash a_2 : \tau \qquad \{y : \tau_2\} \vdash a_3 : \tau}{\emptyset \vdash (\texttt{match}\ v\ \texttt{with}\ \texttt{V}\ x \to a_2 \mid \texttt{E}\ y \to a_3) : \tau}$$

D'après la remarque précédente, ou bien $v = \texttt{V}\ v_1$, ou bien $v = \texttt{E}\ v_2$. Dans le premier cas, a se réduit par la règle (match$_1$), et dans le second cas a se réduit aussi par la règle (match$_2$). D'où le résultat.

Question 4 Oui, le typage de ML_r est sûr, et on le montre en appliquant la preuve générale de sûreté du cours. L'hypothèse (H0) est trivialement vraie.

Nous avons vérifié l'hypothèse (H1) pour les réductions sur le `match`. Le cours la montre pour les règles (δ_+), (δ_{fst}), et (δ_{snd}). Pour $(\delta_/)$, c'est la même preuve que pour (δ_+), en prenant bien sûr $TC(/) = TC(+) = \texttt{int} \times \texttt{int} \to \texttt{int}$. Enfin, (H1) est trivialement vraie pour (δ_{boucle}) car l'expression se réduit en elle-même.

Il reste à vérifier (H2'). C'est déjà fait pour le `match`, et c'est trivial pour les opérateurs `V` et `E`, car `V` v et `E` v sont des valeurs. Le cours l'a montrée pour `+`, `fst` et `snd`. Enfin, `/` appliqué à une paire d'entiers peut toujours se réduire, soit par $(\delta_/)$, soit par (δ_{boucle}). Tout va bien.

Si (δ_{boucle}) est omise, on se retrouve avec le terme $/(1, 0)$ qui est bien typé, n'est pas une valeur, mais ne se réduit pas; la sûreté du typage n'est plus vraie.

Question 5

$$[\![a]\!] = \texttt{V}(\texttt{fun}\ x \to [\![1]\!]) = \texttt{V}(\texttt{fun}\ x \to \texttt{V}\ 1)$$

$$\begin{array}{rl}[\![b]\!] = & \texttt{match}\ \ \texttt{match V}\ c \\ & \qquad\qquad \texttt{with V}\ x \to \texttt{E}\ x \\ & \qquad\qquad\quad \mid\ \texttt{E}\ y \to \texttt{E}\ y \\ & \texttt{with V}\ x \to \texttt{V}\ x \\ & \quad \mid\ \texttt{E}\ y \to \texttt{V}\ 1\end{array}$$

En réduisant les deux `match`, on obtient: $[\![b]\!] \xrightarrow{*} \texttt{V}\ 1$, ce qui correspond bien à l'intuition que l'exception a été rattrapée et la valeur 1 renvoyée normalement.

Question 6 Pour l'application $a_1\ a_2$, si l'évaluation de a_1 lève une exception e, l'évaluation de l'application $a_1\ a_2$ lève aussi la même exception. C'est bien le cas dans la traduction, car alors $[\![a_1]\!]$ s'évalue en `E` e, donc le `match` le plus externe dans $[\![a_1\ a_2]\!]$ se réduit en sa branche `E`, qui renvoie `E` e comme résultat de l'application.

Si l'évaluation de a_1 termine normalement mais celle de a_2 lève une exception e, l'évaluation de $a_1\ a_2$ lève aussi l'exception e. Dans la traduction, $[\![a_1]\!]$ s'évalue en `V` v et $[\![a_2]\!]$ en `E` e. Donc, le `match` le plus externe se réduit en sa branche `V`, et le `match` interne en sa branche `E`, qui renvoie `E` e comme résultat de l'application. C'est le comportement correct.

Si a_1 et a_2 s'évaluent sans lever d'exceptions, la première en une fonction f et la seconde en une valeur v, l'évaluation de $a_1\ a_2$ se poursuit en évaluant f appliquée à v. Cela peut ou bien renvoyer une valeur, ou bien lever une exception suivant le comportement de f. Dans la traduction, $[\![a_1]\!]$ s'évalue en `V` f et $[\![a_2]\!]$ en `V` v. Les deux **`match`** se réduisent en leurs branches `V`, et on finit par renvoyer le résultat de $f\ v$. Ce résultat sera ou bien un `V` ou bien un `E` suivant que f lève ou non une exception. C'est le comportement attendu.

Pour la construction **`try`** a_1 **`with`** $y \to a_2$, si a_1 s'évalue sans lever d'exception en une valeur v, le **`try...with`** tout entier doit s'évaluer en v. Dans la traduction, $[\![a_1]\!]$ s'évalue en `V` v, la première branche du **`match`** est sélectionnée et renvoie `V` v, ce qui correspond bien.

Si a_1 lève l'exception e, alors a_2 est évaluée et son résultat (valeur ou exception) est celui de l'expression **`try...with`** tout entière. Dans la traduction, $[\![a_1]\!]$ s'évalue en `E` e, le **`match`** se réduit en sa deuxième branche, qui évalue $[\![a_2]\!]$ et renvoie son résultat (`V` ou `E`) comme résultat du **`try...with`**. Encore une fois, cela traduit correctement la sémantique du **`try...with`**.

Question 7

$[\![/]\!]$ `= fun x → if snd x = 0 then E divzero else V(/(fst x, snd x))`

(On "renvoie" l'exception `divzero` si le diviseur est nul, et sinon on calcule le résultat de la division et on le renvoie comme valeur normale de résultat. Notez que dans la branche `else`, `snd x` ne vaut jamais 0, et donc le cas de bouclage de la division dans ML_r ne se produit pas.)

Question 8 On raisonne par cas sur la valeur v. Si $v = c$, la traduction simplifiée $\mathcal{S}([\![v]\!])$ est `V` c, qui est bien une valeur de la forme annoncée. Si $v = op$ ou $v =$ **`fun`** $x \to a$, la traduction simplifiée est de la forme `V(`**`fun`** $\ldots$), qui est aussi une valeur de la forme annoncée. Enfin, pour le cas $v = (v_1, v_2)$, on fait une petite récurrence structurelle sur la valeur. Par hypothèse de récurrence, $\mathcal{S}([\![v_1]\!]) =$ `V` v_1' et $\mathcal{S}([\![v_2]\!]) =$ `V` v_2'. Donc, la traduction de v,

$$\begin{array}{l}
\texttt{match}\ [\![v_1]\!] \\
\texttt{with V}\ x_1 \to \ \ \texttt{match}\ [\![v_2]\!] \\
\qquad\qquad\qquad\ \texttt{with V}\ x_2 \to \texttt{V}(x_1, x_2) \\
\qquad\qquad\qquad\qquad \mid\ \texttt{E}\ y_2 \to \texttt{E}\ y_2 \\
\quad \mid\ \texttt{E}\ y_1 \to \texttt{E}\ y_1
\end{array}$$

se simplifie en

$$\begin{array}{l}
\texttt{match V}\ v_1' \\
\texttt{with V}\ x_1 \to \ \ \texttt{match V}\ v_2' \\
\qquad\qquad\qquad\ \texttt{with V}\ x_2 \to \texttt{V}(x_1, x_2) \\
\qquad\qquad\qquad\qquad \mid\ \texttt{E}\ y_2 \to \texttt{E}\ y_2 \\
\quad \mid\ \texttt{E}\ y_1 \to \texttt{E}\ y_1
\end{array}$$

qui se simplifie finalement en $\texttt{V}(v_1', v_2')$. C'est bien une valeur de ML_r de la forme annoncée.

Question 9 **Cas** de la réduction (β_{fun}).

$$\begin{array}{rcl}
\mathcal{S}([\![(\texttt{fun}\ x \to a)\ v]\!]) & = & \mathcal{S}(\ \texttt{match V(fun}\ x \to [\![a]\!])\\
& & \quad \texttt{with V}\ x_1 \to\ \ \texttt{match}\ [\![v]\!]\\
& & \qquad\qquad\qquad \texttt{with V}\ x_2 \to x_1\ x_2\\
& & \qquad\qquad\qquad\quad \mid\ \texttt{E}\ y_2 \to \texttt{E}\ y_2\\
& & \qquad \mid\ \texttt{E}\ y_1 \to \texttt{E}\ y_1\ )\\
& = & \mathcal{S}(\ \texttt{match}\ [\![v]\!]\\
& & \quad \texttt{with V}\ x_2 \to (\texttt{fun}\ x \to [\![a]\!])\ x_2\\
& & \qquad \mid\ \texttt{E}\ y_2 \to \texttt{E}\ y_2\ )\\
& = & \mathcal{S}(\ \texttt{match V}\ \ \overline{v}\\
& & \quad \texttt{with V}\ x_2 \to (\texttt{fun}\ x \to [\![a]\!])\ x_2\\
& & \qquad \mid\ \texttt{E}\ y_2 \to \texttt{E}\ y_2\ )\\
& = & \mathcal{S}((\texttt{fun}\ x \to [\![a]\!])\ \overline{v})\\
& = & (\texttt{fun}\ x \to \mathcal{S}([\![a]\!]))\ \overline{v}
\end{array}$$

(pour la dernière égalité, remarquons qu'il n'y a rien à simplifier ni dans $\texttt{fun}\ x$, ni dans $\overline{v}$.) D'autre part, par le lemme de commutation:

$$\mathcal{S}([\![a\{x \leftarrow v\}]\!]) = (\mathcal{S}([\![a]\!]))\{x \leftarrow \overline{v}\}$$

Et on a bien $(\texttt{fun}\ x \to \mathcal{S}([\![a]\!]))\ v \to (\mathcal{S}([\![a]\!]))\{x \leftarrow \overline{v}\}$ par la réduction (β_{fun}) de ML_r. C'est le résultat attendu.

Cas de la réduction (try_1).

$$\begin{array}{rcl}
\mathcal{S}([\![\texttt{try}\ v\ \texttt{with}\ y \to a]\!]) & = & \mathcal{S}(\ \texttt{match}\ [\![v]\!]\\
& & \quad \texttt{with V}\ x \to \texttt{V}\ x\\
& & \qquad \mid\ \texttt{E}\ y \to [\![a]\!]\ )\\
& = & \mathcal{S}(\ \texttt{match V}\ \overline{v}\\
& & \quad \texttt{with V}\ x \to \texttt{V}\ x\\
& & \qquad \mid\ \texttt{E}\ y \to [\![a]\!]\ )\\
& = & \mathcal{S}(\texttt{V}\ \overline{v})\\
& = & \mathcal{S}([\![v]\!])
\end{array}$$

C'est le résultat attendu.

Cas de la réduction (try_2).

$$\begin{array}{rcl}
\mathcal{S}([\![\texttt{try raise}\ v\ \texttt{with}\ y \to a]\!]) & = & \mathcal{S}(\ \texttt{match}\ [\![\texttt{raise}\ v]\!]\\
& & \quad \texttt{with V}\ x \to \texttt{V}\ x\\
& & \qquad \mid\ \texttt{E}\ y \to [\![a]\!]\ )\\
& = & \mathcal{S}(\ \texttt{match}\ \ \texttt{match V}\ \overline{v}\\
& & \qquad\qquad \texttt{with V}\ x_1 \to \texttt{E}\ x_1\\
& & \qquad\qquad\quad \mid\ \texttt{E}\ y_1 \to \texttt{E}\ y_1\\
& & \quad \texttt{with V}\ x \to \texttt{V}\ x\\
& & \qquad \mid\ \texttt{E}\ y \to [\![a]\!]\ )
\end{array}$$

$$
\begin{array}{ll}
= & \texttt{match E } \overline{v} \\
& \texttt{with V } x \to \texttt{V } x \\
& \quad \mid \texttt{ E } y \to \mathcal{S}([\![a]\!]) \\
\to & \mathcal{S}([\![a]\!])\{y \leftarrow \overline{v}\} \text{ par la règle (match}_2\text{)}
\end{array}
$$

Ce dernier terme étant égal à $\mathcal{S}([\![a\{y \leftarrow \overline{v}\}]\!])$ par le lemme de commutation, nous obtenons le résultat annoncé.

Cas Réduction (propagation$_1$). Pour le membre droit de la règle, nous avons:

$$
\begin{array}{lll}
\mathcal{S}([\![\texttt{raise } v]\!]) & = & \mathcal{S}(\texttt{ match V } \overline{v} \\
& & \quad \texttt{with V } x_0 \to \texttt{E } x_0 \\
& & \qquad \mid \texttt{ E } y_0 \to \texttt{E } y_0 \;) \\
& = & \mathcal{S}(\texttt{E } \overline{v})
\end{array}
$$

et pour le membre gauche:

$$
\begin{array}{lll}
\mathcal{S}([\![(\texttt{raise } v)\ a]\!]) & = & \mathcal{S}(\texttt{ match E } \overline{v} \\
& & \quad \texttt{with V } x_1 \to \; \texttt{match } [\![a]\!] \\
& & \qquad\qquad \texttt{with V } x_2 \to x_1\ x_2 \\
& & \qquad\qquad\quad \mid \texttt{ E } y_2 \to \texttt{E } y_2 \\
& & \qquad \mid \texttt{ E } y_1 \to \texttt{E } y_1 \;) \\
& \to & \mathcal{S}(\texttt{E } \overline{v}) \text{ par la règle (match}_2\text{)}
\end{array}
$$

D'où l'égalité attendue.

Question 10

- $\mathcal{S}([\![1]\!]) = \texttt{V } 1$ a le type principal $(\texttt{int}, \alpha)\ \texttt{res}$.
- $\mathcal{S}([\![\texttt{raise divzero}]\!]) = \texttt{E divzero}$ a le type principal $(\alpha, \texttt{exn})\ \texttt{res}$.
- $\mathcal{S}([\![\texttt{try raise divzero with } y \to 1]\!]) = \texttt{match E divzero with V } x \to \texttt{V } x \mid \texttt{E } y \to \texttt{V } 1$ a le type principal $(\texttt{int}, \alpha)\ \texttt{res}$.

Question 11 Supposons $a \xrightarrow{*} \texttt{raise } v$. D'après la question 9, chaque réduction dans ML_e correspond à 0 ou 1 réductions sur les traductions simplifiées dans ML_r. Donc, $\mathcal{S}([\![a]\!]) \xrightarrow{*} \mathcal{S}([\![\texttt{raise } v]\!])$ dans ML_r.

D'autre part, les réductions dans ML_r préservent le typage (question 2), donc si $\emptyset \vdash \mathcal{S}([\![a]\!]) : (\tau, \alpha)\ \texttt{res}$, alors $\emptyset \vdash \mathcal{S}([\![\texttt{raise } v]\!]) : (\tau, \alpha)\ \texttt{res}$.

Or, $\mathcal{S}([\![\texttt{raise } v]\!]) = \texttt{E } \overline{v}$. On aurait donc la dérivation de typage suivante dans ML_r:

$$
\dfrac{\dfrac{\alpha \to (\tau, \alpha)\ \texttt{res} \leq TC(\texttt{E})}{\emptyset \vdash \texttt{E} : \alpha \to (\tau, \alpha)\ \texttt{res}} \qquad \emptyset \vdash \overline{v} : \alpha}{\emptyset \vdash \texttt{E } \overline{v} : (\tau, \alpha)\ \texttt{res}}
$$

Mais ceci est impossible car $\overline{v}$ est une valeur de ML_r, et il n'existe pas de valeur ayant le type α: les seuls types possibles pour une valeur sont `int` ou `exn` (si c'est une constante), un type flèche (si c'est un opérateur ou une fonction), un type produit (si c'est une paire de valeurs), ou un type (τ_1, τ_2) `res` (si c'est un `V` ou un `E`).

D'où contradiction: l'expression a ne peut pas se réduire en `raise` v.

Question 12 L'algorithme consiste tout simplement à traduire et simplifier l'expression a, puis à inférer le type principal de $\mathcal{S}([\![a]\!])$ dans ML_r en utilisant l'algorithme W. Si ce type principal est de la forme (τ, α) `res`, alors renvoyer `faux`. Sinon, renvoyer `vrai`.

Preuve de correction: supposons que l'algorithme renvoie `faux`. Alors, l'algorithme W renvoie (τ, α) `res` comme type principal de $\mathcal{S}([\![a]\!])$. Puisque l'algorithme W est correct, cela veut dire que $\emptyset \vdash \mathcal{S}([\![a]\!]) : (\tau, \alpha)$ `res`. D'après la question 11, cela entraîne que a ne peut pas se réduire en `raise` v.