Programmation fonctionnelle et systèmes de types (MPRI 2-4-2)

Problème 1

13 février 2007

Le but de ce problème est d'étudier certaines optimisations de compilation que l'on peut effectuer par réécriture sur le programme source. Nous montrerons que ces optimisations sont correctes, c'est-à-dire qu'elles préservent le comportement du programme.

1 Le langage source

Le langage source considéré est le λ-calcul en appel par valeur enrichi de constantes entières (notées N) :

Termes : $a, b, c ::= N \mid x \mid \lambda x.a \mid a\ b$

Valeurs : $v ::= N \mid \lambda x.a$

On notera $\texttt{let}\ x = a\ \texttt{in}\ b$ le terme $(\lambda x.b)\ a$.

La relation de réduction $\to$ définissant la sémantique de ce langage est la suivante :

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

2 Règles d'optimisation

On définit les optimisations qui nous intéressent par les règles de réécriture suivantes :

(O_1)	$(\lambda x.a)\ b$	$\hookrightarrow$	$a[x \leftarrow b]$ si b est une valeur v ou une variable y
(O_2)	$\texttt{let}\ x = (\texttt{let}\ y = a\ \texttt{in}\ b)\ \texttt{in}\ c$	$\hookrightarrow$	$\texttt{let}\ y = a\ \texttt{in}\ (\texttt{let}\ x = b\ \texttt{in}\ c)$ si $x \neq y$ et y non libre dans c

Notez bien la différence d'intention entre les relations $\to$ et $\hookrightarrow$: la relation $\to$ décrit ce qui se passe pendant l'exécution du programme ; la relation $\hookrightarrow$ décrit les optimisations possibles effectuées pendant la compilation du programme.

Question 1 On considère le programme

$$P = \texttt{let}\ x = (\texttt{let}\ y = a\ \texttt{in}\ \lambda z.b)\ \texttt{in}\ x\ \texttt{true}$$

En quel programme P' le programme P peut-il s'optimiser en appliquant une ou plusieurs fois les règles O_1 et O_2 ? Intuitivement, pourquoi P' s'exécute-t'il plus efficacement que P ?

3 Correction des optimisations appliquées en tête de programme

Dans cette section, on applique les règle d'optimisations uniquement en tête des programmes. Autrement dit, le compilateur peut remplacer le programme a par le programme a' si $a \hookrightarrow a'$. Nous voulons montrer que a et a' ont le même comportement observable, c'est-à-dire :

- si a diverge (se réduit infiniment par $\to$), alors a' diverge aussi ;
- si $a \xrightarrow{*} N$ alors $a' \xrightarrow{*} N$.

Question 2 Montrer que si $a \to a'$, alors a et a' ont le même comportement observable.

Question 3 En déduire que si a est clos (sans variables libres) et si $a \hookrightarrow a'$ par la règle O_1, alors a et a' ont le même comportement observable.

Question 4 Est-ce que ce résultat reste vrai si on enlève la restriction "b est une valeur ou une variable" dans la règle O_1 ?

Question 5 Supposons $a \hookrightarrow a'$ par la règle O_2. Quelle est la forme générale d'une séquence de réductions $\to$ issue de a ? D'une séquence de réductions issue de a' ? En déduire que a et a' ont le même comportement observable.

4 Correction de l'optimisation O_1 appliquée n'importe où dans le programme

Dans cette section, nous permettons maintenant au compilateur d'appliquer les règles d'optimisation en un point quelconque du programme et non plus seulement en tête. Autrement dit, si $a \hookrightarrow a'$, et si C est un contexte quelconque (pas nécessairement un contexte d'évaluation), le compilateur peut remplacer le programme $C[a]$ par le programme optimisé $C[a']$. Cela permet notamment d'optimiser à l'intérieur des fonctions :

$$\texttt{let } f = \lambda x.\ (\texttt{let } y = 0 \texttt{ in } x) \texttt{ in } a$$

peut ainsi être optimisé en

$$\texttt{let } f = \lambda x.x \texttt{ in } a$$

Nous allons prouver qu'une telle optimisation préserve le comportement observable du programme dans le cas particulier de la règle O_1. Nous définissons une généralisation $\rightsquigarrow$ de la règle O_1 par les règles d'inférence suivantes :

$$\frac{a[x \leftarrow b] \rightsquigarrow c \qquad b \text{ est une valeur ou une variable}}{(\lambda x.a)\ b \rightsquigarrow c}\ (1)$$

$$N \rightsquigarrow N\ (2) \qquad\qquad x \rightsquigarrow x\ (3)$$

$$\frac{a \rightsquigarrow a'}{\lambda x.a \rightsquigarrow \lambda x.a'}\ (4) \qquad\qquad \frac{a \rightsquigarrow a' \qquad b \rightsquigarrow b'}{a\ b \rightsquigarrow a'\ b'}\ (5)$$

Intuitivement, la relation $\rightsquigarrow$ permet d'appliquer l'optimisation O_1 zéro, une ou plusieurs fois n'importe où dans le programme (règles (4) et (5)). Elle permet aussi d'enchaîner un nombre fini d'optimisations (règle (1)).

On admettra le lemme de substitution suivant :

Si $a \rightsquigarrow a'$ et $b \rightsquigarrow b'$ et b est une valeur ou une variable, alors $a[x \leftarrow b] \rightsquigarrow a'[x \leftarrow b']$.

Question 6 Montrer que si $a \to a'$ et $a \rightsquigarrow b$, alors
- ou bien il existe b' tel que $b \to b'$ et $a' \rightsquigarrow b'$;
- ou bien $a' \rightsquigarrow b$, et de plus la dérivation de $a' \rightsquigarrow b$ est strictement plus petite que celle de $a \rightsquigarrow b$.

On procédera par récurrence sur la dérivation de $a \to a'$ et par cas sur les règles utilisées pour conclure $a \to a'$ et $a \rightsquigarrow b$. On traitera les cas $(\lambda x.a)\ v \to a[x \leftarrow v]$ et $a\ c \to a'\ c$ si $a \to a'$, et on admettra le résultat dans les autres cas.

Question 7 Supposons $a \rightsquigarrow b$. Déduire de la question précédente que si $a \xrightarrow{*} N$, alors $b \xrightarrow{*} N$; et que si a diverge, alors b diverge aussi.

Question 8 Supposon $a \hookrightarrow b$ par la règle O_1. Soit C un contexte quelconque. Montrer que $C[a] \rightsquigarrow C[b]$. En déduire que le programme initial $C[a]$ et le programme transformé $C[b]$ ont le même comportement observable.

5 Optimisation et transformation CPS

Dans cette section, nous illustrons l'affirmation répandue selon laquelle "effectuer une transformation CPS pendant la compilation permet davantage d'optimisations". Plus précisément, nous allons voir que l'optimisation O_2 devient inutile si le compilateur utilise la transformation CPS : l'optimisation O_1, appliquée après la transformation CPS, donne les mêmes résultats que O_2 appliquée avant la transformation CPS.

On rappelle la transformation CPS $[\![a]\!]$ d'un terme a :

$$\begin{aligned}
[\![N]\!] &= \lambda k.\ k\ N \\
[\![x]\!] &= \lambda k.\ k\ x \\
[\![\lambda x.a]\!] &= \lambda k.\ k\ (\lambda x.\ [\![a]\!]) \\
[\![a\ b]\!] &= \lambda k.\ [\![a]\!]\ (\lambda y.\ [\![b]\!]\ (\lambda z.\ y\ z\ k))
\end{aligned}$$

Question 9 Montrer que si $a \hookrightarrow a'$ par la règle O_2, alors il existe un terme b tel que $[\![a]\!] \xhookrightarrow{*} b$ et $[\![a']\!] \xhookrightarrow{*} b$ par la règle O_1.