Université Montpellier II — Master d'Informatique
UEMINM 121 Evaluation des Langages Applicatifs

R. Ducournau – M. Lafourcade

juillet 2006 — 2 heures
Documents autorisés : notes de cours, polys, pas de livre.

On se place dans le cadre du méta-évaluateur du cours, dit de première génération.

Environnements dynamiques et lexicaux

Dans les langages de programmation, on caractérise les variables par un notion spatiale de portée — le sous-ensemble du code où la variable est connue — et une notion temporelle de durée de vie. La variable est *liée* à une valeur à un endroit donné dans le programme et à un moment donnée de l'exécution du programme — en général lors de l'activation de la fonction qui l'introduit —, et la *liaison* ainsi produite est regroupée avec d'autres dans un *environnement* qui perdure jusqu'à la fin de la durée de vie de la variable, ou jusqu'à ce que la variable ne soit plus utilisée.

Environnements lexicaux : espace limité et temps illimité

Les évaluateurs habituels — et le méta-évaluateur vu en cours en fait partie — utilisent en général des *environnements lexicaux*, dans lesquels la portée d'une variable est réduite à la forme syntaxique qui l'introduit mais sa durée de vie est illimitée car l'environnement peut être capturé dans une fermeture.

Exemple Ainsi, dans les deux fonctions suivantes :

```
(defun foo (x)
       (bar (* x x)))

(defun bar (y)
       (* x y))
```

la fonction `bar` est incorrecte, **quelle que soit la façon dont on l'utilise**, car elle utilise une variable `x` qui n'est pas introduite dans `bar`, mais dans une autre fonction `foo`.

Question 1

On se place d'abord dans le cadre du méta-évaluateur du cours :

1. Montrer comment se fait le passage de paramètres en environnement lexical : reprenez uniquement le cas des fonctions méta-définies ainsi que celui des λ-expressions.
   Expliciter les différences.
2. Montrer en détaillant les environnements pourquoi les évaluations de `(bar 4)` et de `(foo 3)` entraînent une exception.
3. Montrer a contrario ce qui se passerait si l'appel à `bar` dans `foo` était remplacé par l'application de la λ-expression qui définit `bar`.

Environnements dynamique : espace illimité et temps limité

Dans un *environnement dynamique*, la portée d'une variable est le progamme tout entier, mais la variable n'est accessible que pendant la durée de l'activation d'une fonction qui introduit la variable. Ainsi, l'évaluation de `(foo 3)` retournera 27, mais l'évaluation, au *top level*, de `(bar 4)` provoquera une exception, la variable `x` n'étant pas dans l'environnement dynamique de cette évaluation.

Question 2

On va maintenant modifier le méta-évaluateur du cours pour remplacer les environnement lexicaux par des environnement dynamiques :

1. modifiez d'abord le cas des fonctions méta-définies.

2. Faut-il faire une modification dans le cas des λ-expressions ? si oui laquelle ? sinon pourquoi ?
3. Montrer en détaillant les environnements pourquoi les évaluations de `(bar 4)` et de `(foo 3)` entraînent, ou pas, une exception.

NB. Dans cette question, détaillez ce qui diffère et expliquez succinctement ce qui ne change pas. Ne considérez pas maintenant le cas des fermeturs (cf. question 5).

Question 3

Comparer les comportements de l'évaluateur selon que les environnements sont lexicaux ou dynamiques.

1. Est-il possible que, dans les deux cas, l'évaluation se termine normalement mais en retournant des valeurs différentes ?
2. Donnez des arguments en faveur de la liaison lexicale.

Question 4

Dans le cours, le "méta-évaluateur de deuxième génération" était basé sur une transformation statique du code LISP de telle sorte que les environnements lexicaux soient implémentés par des tableaux, les variables étant remplacées par leur position dans l'environnement. Cette transformation permet un gain appréciable de performance.

Une telle transformation serait-elle possible avec des environnements dynamiques ? Proposer des solutions.

Question 5

La durée de vie illimitée des environnements lexicaux s'obtient par la capture d'un environnement par une fermeture.

1. Montrer que les environnements dynamiques permettent en partie de se passer de la notion de fermeture.

   Examiner pour cela les exemples suivants :

```
(defun foo1 (x y z)
       (labels ((bar (w)
                     (if (eq x w)
                          y
                       (cons (bar (car w)) (bar (cdr w))))))
           (bar z)))
```

   Dans `foo1`, la fonction locale définie par `labels` doit-elle être une fermeture, c'est-à-dire doit-elle capturer l'environnement de sa définition (les variables `x`, `y` et `z`) ?

```
(defun foo2 (x y)
       (labels ((bar (w)
                     (if (eq x w)
                          y
                       (cons (bar (car w)) (bar (cdr w))))))
           #'bar))
```

   Comparer l'effet de `(foo1 1 2 '((1 . 3) . 1))` et de `(apply (foo2 1 2) '((1 . 3) . 1))` suivant que les environnements sont lexicaux ou dynamiques.

2. Comment pourrait-on définir des fermetures avec des environnements dynamiques ? Examiner le problème sous l'angle de la sémantique (qu'est-ce que cela apporterait ?) et sous celui de l'implémentation (est-ce implémentable, et comment ?)