UFR 919 Sciences de l'ingénieur
Master Informatique – Spécialité STL
Analyse de Programmes et Sémantique (APS)

TD8 — Sémantique dénotationnelle et objets

Jacques Malenfant, Olena Rogovchenko

1 Représentation fonctionnelle des structures de données

1.1 Paires

On définit la fonction *make-pair* de la façon suivante :

$$
\begin{array}{l}
\textit{make-pair} = (\textbf{fix}\ \lambda f. \\
\qquad \lambda a.\lambda b.\lambda cmd. \\
\qquad\quad \textbf{if}\ cmd = \texttt{1st}\ \textbf{then}\ a \\
\qquad\quad \textbf{else if}\ cmd = \texttt{2nd}\ \textbf{then}\ b \\
\qquad\quad \textbf{else if}\ cmd = \texttt{set-1st}\ \textbf{then}\ \lambda v.((f\ v)\ b) \\
\qquad\quad \textbf{else if}\ cmd = \texttt{set-2nd}\ \textbf{then}\ \lambda v.((f\ a)\ v) \\
\qquad\quad \textbf{else}\ error)
\end{array}
$$

Que donnent les appels suivants :

— ((*make-pair* 3) 4),
— (((*make-pair* 3) 4) `1st`), et
— ((((*make-pair* 3) 4) `set-1st`) 7)

1.2 Réels

En s'inspirant de cette définition, définissez une fonction *make-real* avec les opérations `get` pour récupérer la valeur, `add` pour additionner deux réels et `sqrt` pour retourner la racine carrée. Pour le calcul de la racine carrée, vous appliquerez un algorithme de recherche dichotomique implanté à l'aide d'un point fixe, comme il se doit... (vous pouvez supposer que vous avez une fonction *abs* qui retourne la valeur absolue d'un réel).

1.3 Listes

En s'inspirant des cas précédents, définissez la fonction *make-list* avec les opérations `cons`, `car` et `cdr`. Définissez également la structure de base *empty-list* et rajouter le test `isEmpty` à la structure. Rajoutez enfin l'opération `member`.

2 Le test « instanceof »

Dans l'implantation courante de BOPL, soit pc une instance de la classe **PointColore** qui hérite de la classe **Point**, on a :

— $\mathscr{E}[\![pc$ `instanceof` **PointColore**$]\!]\rho\sigma o \rightarrow \langle true, \sigma, o\rangle$
— $\mathscr{E}[\![pc$ `instanceof` **Point**$]\!]\rho\sigma o \rightarrow \langle false, \sigma, o\rangle$

Comment modifier la sémantique d'`instanceof` pour que l'expression « x `instanceof` Y » soit vraie dans le cas où x est l'instance d'une sous-classe de Y ?

3 Héritage multiple

Certains langages (C++, OCaml, Python...) permettent l'héritage multiple. On veut étendre BOPL pour permettre à une classe d'avoir plusieurs superclasses. Proposer un nouvel algorithme pour la sémantique d'un appel de méthode, en tenant compte de l'héritage multiple.

Quelles sont les modifications nécessaires à la représentation des classes, des objets et des méthodes pour la mise en oeuvre de cette extension ?

4 Partie TME

4.1 Représentation fonctionnelle des structures de données

Réalisez en Scheme la représentation fonctionnelle des réels et des listes.

4.2 Extensions à BOPL

Étendez l'implantation de la sémantique dénotationnelle de BOPL avec le nouveau test `instanceof`.