TD 7: Élimination des références et des exceptions dans MINI-ML

Ioana Pasca, Marc Lasson

On rappelle la syntaxe de MINI-ML vue en cours :

$$
\begin{array}{rcl}
e, e_1, e_2, \ldots & := & x \\
& | & 0 \quad | \quad 1 \quad \cdots \\
& | & \textbf{true} \quad | \quad \textbf{false} \\
& | & \textbf{fun}\ x \to e \\
& | & (e_1\, e_2) \\
& | & \textbf{let}\, x = e_1\, \textbf{in}\, e_2 \\
& | & e_1 + e_2 \quad | \quad e_1 * e_2 \quad \cdots \\
& | & e_1 = e_2 \quad | \quad e_1 < e_2 \quad \cdots \\
& | & \textbf{not}\, e_1 \quad | \quad e_1\, \textbf{and}\, e_2 \quad \cdots \\
& | & \textbf{if}\, e_1\, \textbf{then}\, e_2\, \textbf{else}\, e_3
\end{array}
$$

Question 1. Retrouvez la sémantique à grand pas pour cette syntaxe que vous avez vue en cours.

Question 2. Donnez un exemple de programme qui ne s'évalue pas sur une valeur. Quel outil pourrait-on utiliser pour s'interdire ce genre de programmes ?

Question 3. Expliquez en quoi le point-virgule de `caml` peut-être vu comme du sucre syntaxique.

Question 4. Étendez la syntaxe et la sémantique pour pouvoir manipuler des couples de valeurs.

Question 5. Proposez une implémentation des listes avec des couples, puis programmez les fonctions `head`,`tail`,`is_nil`.

On rajoute la constante **fix** à la syntaxe et on étend la sémantique avec la règle suivante :

$$\frac{e \Downarrow (\textbf{fun}\, x \to e') \qquad e'[\textbf{fix}\, e/x] \Downarrow v}{\textbf{fix}\, e \Downarrow v}$$

Question 6. Est-ce que vous voyez comment la constante **fix** nous permet de simuler la définition des fonctions récursives de `caml` ? Déduisez-en une implémentation dans notre langage de la fonction qui retourne la longueur d'une liste.

Question 7. Cette constante n'est pas dans la bibliothèque standard de `caml`, sauriez-vous la programmer ? Quel serait son type ?

Question 8 (Difficile, à la maison). Comme on se trouve dans un cadre non typé, il se trouve qu'il existe des termes qui ne contiennent pas la constante **fix** et qui vérifient pourtant la règle de **fix**. Sauriez-vous en trouver un ?

On se donne un ensemble $\mathbb{A}$ infini $\{r_1, r_2, \ldots\}$ *d'adresses* et on étend la syntaxe du langage avec les constructions suivantes :

$$\begin{array}{rcl} e, e_1, e_2, \ldots & := & \ldots \\ & | & e_1 := e_2 \\ & | & !e \\ & | & \mathbf{ref}\, e \end{array}$$

On étend l'ensemble $\mathbb{V}$ des valeurs avec les adresses et on définit l'ensemble Σ des *états* comme étant l'ensemble des fonctions à support fini de $\mathbb{A} \longrightarrow \mathbb{V}$.

Question 9. Expliquez comment vous pourriez étendre la sémantique pour décrire le comportement des références.

Question 10. Implémentez dans notre mini-ML étendu sans référence toutes les primitives pour manipuler les fonctions à support fini inclus dans $\mathbb{N}$:

- *get* $x\, l$: récupère la valeur associée à x dans l,
- *update* $x\, v\, l$: remplace la valeur associée à x par v dans l,
- *new* l : retourne un entier qui n'est associé à rien dans l.

Question 11. En déduire une compilation $\cdot \mapsto [\![\cdot]\!]$ telle que

$$\langle \varrho, e\rangle \Downarrow_{ref} \langle \varrho', v\rangle \text{ implique } ([\![e]\!]\, [\![\varrho]\!]) \Downarrow ([\![v]\!], [\![\varrho']\!])$$

où $[\![e]\!]$ est une expression sans références (ie. sans !, **ref**, ni :=) et où $[\![v]\!], [\![\varrho]\!], [\![\varrho']\!]$ sont des valeurs sans adresses. Esquissez ensuite une preuve que ce procédé de compilation est correct.

On rajoute maintenant dans la syntaxe vue en cours des mécanismes d'exceptions (on ne considère plus les extensions définies lors des précédentes questions).

$$\begin{array}{rcl} e, e_1, e_2, \ldots & := & \ldots \\ & | & \mathbf{raise}\, e_2 \\ & | & \mathbf{try}\ e_1\ \mathbf{with}\, x \to e_2 \end{array}$$

Question 12. Concevez une sémantique à grand-pas pour gérer les exceptions.

On notera $\Downarrow_{exn}$ la sémantique des comportements exceptionnels.

Question 13. Définissez une compilation $\cdot \mapsto |\cdot|$ qui vérifie

$$\begin{array}{rl} e \Downarrow v \wedge (f\, v) \Downarrow u & \Rightarrow |e|\, f\, g \Downarrow u \\ e \Downarrow_{exn} v \wedge (g\, v) \Downarrow u & \Rightarrow |e|\, f\, g \Downarrow u \end{array}$$

où f, g et $|e|$ sont des expressions sans exceptions (ie. ne contenant ni **raise** ni **try** $\cdots$ **with** $\cdots$).