Traduction des langages
Interprétation d'un sous-ensemble de Caml : mini-ML

Objectifs : Le but de ce TP est d'apprendre à écrire un interpréteur pour un langage (mini-ML dont la grammaire est rappelée dans la section 2) à partir de l'arbre abstrait des programmes.

Les règles de la sémantique dynamique qui décrivent l'exécution ont été présentées en TD et sont rappelées dans la section 3.

Lors du dernier TP de programmation fonctionnelle, une partie de cette évaluateur a été écrite en utilisant (massivement) les modules pour illustrer la problématique de l'évolution d'une architecture. Ici, l'architecture est toute autre, car nous supposons notre grammaire du langage connue et figée.

1 À faire

L'arbre abstrait est décrit par le type ast dans le fichier Ast.ml.
Vous testerez votre implantation à chaque étape.

1. **Exécution des expressions arithmétiques**
   La sémantique est déjà implantée. Il s'agit des fonctions ruleBinary, ruleUnary, ruleInteger, ruleTrue et ruleFalse.
2. **Exécution des conditionnelles et définitions de variables**
   Il s'agit des fonctions ruleLet et ruleIf.
3. **Exécution des définitions et appel par valeur de fonctions**
   Il s'agit des fonctions ruleFunction et ruleCallByValue.
   La valeur des paramètres doit être calculée avant l'appel de la fonction (appel par valeur).
   Implanter la sémantique de l'appel par valeur.
4. **Exécution des définitions récursives**
   Il s'agit de la fonction ruleLetRec.
5. **Appel par nom de fonctions**
   Une deuxième sémantique est possible pour les appels de fonctions, l'appel par nom : le calcul des paramètres est suspendu (constructeur de valeur FrozenValue) et l'appel de la fonction est exécuté. La valeur des paramètres sera calculée lors de l'accès au paramètre.
   Implanter la sémantique de l'appel par nom (ruleCallByName).
6. **Appel paresseux de fonctions**
   Lors d'un appel par nom, la valeur du paramètre est calculée à chaque fois que la fonction utilise le paramètre. L'appel paresseux consiste à partager le résultat de cette évaluation.
   Proposer une solution pour implanter l'appel paresseux.

2 Grammaire

$$
\begin{array}{rcl}
\mathit{Expr} & \to & \mathit{Ident} \\
& | & \mathit{Const} \\
& | & \mathit{Expr}\ \mathit{Binaire}\ \mathit{Expr} \\
& | & \mathit{Unaire}\ \mathit{Expr} \\
& | & \texttt{(}\ \mathit{Expr}\ \texttt{)} \\
& | & \texttt{if}\ \mathit{Expr}\ \texttt{then}\ \mathit{Expr}\ \texttt{else}\ \mathit{Expr} \\
& | & \texttt{let}\ \mathit{Ident}\ \texttt{=}\ \mathit{Expr}\ \texttt{in}\ \mathit{Expr} \\
& | & \texttt{fun}\ \mathit{Ident}\ \texttt{->}\ \mathit{Expr} \\
& | & \texttt{(}\mathit{Expr}\ \texttt{(}\ \mathit{Expr}\texttt{))} \\
& | & \texttt{let rec}\ \mathit{Ident}\ \texttt{=}\ \mathit{Expr}\ \texttt{in}\ \mathit{Expr}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{Const} & \to & \mathit{entier} \mid \mathit{booleen} \\
\mathit{Unaire} & \to & \texttt{-} \mid \texttt{!} \\
\mathit{Binaire} & \to & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%} \mid \texttt{\&} \mid \texttt{|} \\
& | & \texttt{==} \mid \texttt{!=} \mid \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=}
\end{array}
$$

3 Sémantique opérationnelle

Constante

$$\gamma \vdash \mathit{entier} \Rightarrow \mathit{entier} \quad \gamma \vdash \mathit{booleen} \Rightarrow \mathit{booleen}$$

Accès à l'environnement

$$\frac{x \in \gamma \quad \gamma(x) = v \quad v \neq fix(\, e\,, \gamma_{def}\,)}{\gamma \vdash x \Rightarrow v}$$

$$\frac{x \in \gamma \quad \gamma(x) = fix(\, e\,, \gamma_{def}\,)}{\gamma \vdash x \Rightarrow \langle\, e\,, \{x \mapsto fix(\, e\,, \gamma_{def}\,)\} :: \gamma_{def}\,\rangle}$$

Opérateur binaire

$$\frac{\gamma \vdash e_1 \Rightarrow v_1 \quad \gamma \vdash e_2 \Rightarrow v_2 \quad v_1 \times v_2 \in dom\, op \quad v = v_1\, op\, v_2}{\gamma \vdash e_1\ op\ e_2 \Rightarrow v}$$

Opérateur unaire

$$\frac{\gamma \vdash e \Rightarrow v \quad v \in dom\, op \quad v' = op\, v}{\gamma \vdash op\ e \Rightarrow v'}$$

Conditionnelle

$$\frac{\gamma \vdash e_1 \Rightarrow \texttt{true} \quad \gamma \vdash e_2 \Rightarrow v}{\gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \Rightarrow v} \qquad \frac{\gamma \vdash e_1 \Rightarrow \texttt{false} \quad \gamma \vdash e_3 \Rightarrow v}{\gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \Rightarrow v}$$

Définition locale

$$\frac{\gamma \vdash e_1 \Rightarrow v_1 \quad \{x \mapsto v_1\} :: \gamma \vdash e_2 \Rightarrow v}{\gamma \vdash \texttt{let}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 \Rightarrow v}$$

Définition de fonction

$$\gamma \vdash \texttt{fun}\ x\ \texttt{->}\ e \Rightarrow \langle\, \texttt{fun}\ x\ \texttt{->}\ e\,,\, \gamma \,\rangle$$

Appel de fonction

$$\frac{\gamma \vdash e_2 \Rightarrow v_2 \quad \gamma \vdash e_1 \Rightarrow \langle\, \texttt{fun}\ x\ \texttt{->}\ e_3\,,\, \gamma_{def} \,\rangle \quad \{x \mapsto v_2\} :: \gamma_{def} \vdash e_3 \Rightarrow v}{\gamma \vdash (e_1\ (\ e_2)) \Rightarrow v}$$

Définition récursive

$$\frac{\{x \mapsto fix(\, e_1\,,\, \gamma\,)\} :: \gamma \vdash e_2 \Rightarrow v}{\gamma \vdash \texttt{let rec}\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 \Rightarrow v}$$

Gestion des erreurs

Il faut ajouter à ces règles, celles d'apparition et de propagation des erreurs.