Spécifications formelles, vérification, validation (HMIN203)

Mars 2016

Master AIGLE

Département Informatique

Faculté des Sciences de Montpellier

Projet Coq : Sémantique formelle d'un langage fonctionnel

1 Présentation

On considère un mini-langage fonctionnel dont on veut spécifier la sémantique formelle. La syntaxe de ce langage est définie dans la figure 1, où l'on définit les expressions, les types, et les valeurs (expressions après évaluation) du langage. On pourra remarquer que les fonctions doivent indiquer le type de leur argument, alors que les fonctions récursives doivent en plus indiquer le type de leur résultat (nous verrons en effet que c'est nécessaire pour les typer correctement). On remarquera également que les valeurs sont soit des entiers, soit des booléens, soit des valeurs fonctionnelles (appelées aussi fermetures) récursives ou non, soit une erreur si on a tenté d'évaluer une expression non appropriée.

Nous allons donner une sémantique opérationnelle naturelle (à grands pas) à ce langage. La relation d'évaluation est de la forme $\Gamma \vdash e \hookrightarrow v$, où Γ est un contexte d'évaluation qui est une liste de couples (x_i, v_i) avec x_i une variable et v_i une valeur, et où e est une expression, et v une valeur. Cette relation se lit « l'expression e s'évalue en v dans le contexte Γ ». Les règles d'évaluation sont sous la forme de règles d'inférence et sont données dans la figure 2. Ces règles doivent être complétées par des règles d'erreurs, que l'on ne donnera pas ici par souci de concision mais qui seront à formaliser dans le projet. Par exemple, pour les opérations arithmétiques binaires, ces règles d'erreurs ont la forme suivante :

$$\frac{\Gamma \vdash e_1 \hookrightarrow v_1 \notin \mathbb{Z}}{\Gamma \vdash e_1 \ \mathsf{op}\ e_2 \hookrightarrow \mathsf{Err}}\ \mathsf{op}_{\mathbb{Z}\mathsf{Err}_1} \qquad \frac{\Gamma \vdash e_1 \hookrightarrow n_{1\mathbb{Z}} \quad \Gamma \vdash e_2 \hookrightarrow v_2 \notin \mathbb{Z}}{\Gamma \vdash e_1 \ \mathsf{op}\ e_2 \hookrightarrow \mathsf{Err}}\ \mathsf{op}_{\mathbb{Z}\mathsf{Err}_2}$$

$$\frac{\Gamma \vdash e_1 \hookrightarrow \mathsf{Err}}{\Gamma \vdash e_1 \ \mathsf{op}\ e_2 \hookrightarrow \mathsf{Err}}\ \mathsf{op}_{\mathbb{Z}\mathsf{Err}_3} \qquad \frac{\Gamma \vdash e_1 \hookrightarrow n_{1\mathbb{Z}} \quad \Gamma \vdash e_2 \hookrightarrow \mathsf{Err}}{\Gamma \vdash e_1 \ \mathsf{op}\ e_2 \hookrightarrow \mathsf{Err}}\ \mathsf{op}_{\mathbb{Z}\mathsf{Err}_4}$$

où $\mathsf{op} \in \{+, -, \times\}$.

e	$::=$	n	Constantes entières
	$\mid$	$\mathsf{true} \mid \mathsf{false}$	Constantes booléennes
	$\mid$	x	Variables
	$\mid$	$-e \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 \times e_2$	Opérations sur les entiers
	$\mid$	$e_1 = e_2 \mid e_1 \neq e_2 \mid e_1 < e_2 \mid e_1 > e_2 \mid e_1 \leq e_2 \mid e_1 \geq e_2$	Relations sur les entiers
	$\mid$	$\mathsf{not}(e) \mid e_1 \ \mathsf{and} \ e_2 \mid e_2 \ \mathsf{or} \ e_2$	Opérations sur les booléens
	$\mid$	$\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3$	Conditionnelle
	$\mid$	$\mathsf{let}\ x : \tau = e_1\ \mathsf{in}\ e_2$	Déclaration locale
	$\mid$	$\mathsf{fun}\ (x\ :\ \tau) \to e$	Fonction
	$\mid$	$\mathsf{recfun}\ f\ (x\ :\ \tau_1)\ :\ \tau_2 \to e$	Fonction récursive
	$\mid$	$e_1\ e_2$	Application

τ	$::=$	int	Type des entiers
	$\mid$	bool	Type des booléens
	$\mid$	$\tau_1 \to \tau_2$	Type des fonctions

v	$::=$	n	Valeurs entières
	$\mid$	$\top \mid \bot$	Valeurs booléennes
	$\mid$	$<\Gamma, \mathsf{fun}\ (x\ :\ \tau) \to e>$	Valeurs des fonctions
	$\mid$	$<\Gamma, \mathsf{recfun}\ f\ (x\ :\ \tau_1)\ :\ \tau_2 \to e>$	Valeurs des fonctions récursives
	$\mid$	Err	Erreur

où Γ est une liste de couples (x, v) avec x une variable et v une valeur.

FIGURE 1 – Expressions, types, et valeurs

Nous allons également pourvoir ce langage d'un système de types. Les types sont donnés dans la figure 1 (entrée τ). La relation de typage est de la forme $\Gamma \vdash e : \tau$, où Γ est un contexte de typage qui est une liste de couples (x_i, τ_i) avec x_i une variable et τ_i un type, et où e est une expression, et τ un type. Cette relation se lit « l'expression e est bien typé et de type τ dans Γ. Les règles de typage sont sous la forme de règles d'inférence et sont données dans la figure 3.

2 Travail à faire

Vous devrez réaliser en Coq la formalisation du langage précédemment décrit. En particulier, vous devrez réaliser les tâches suivantes :

1. Formaliser les expressions, les types, et les valeurs sous la forme de types inductifs. Écrire quelques exemples de programmes, dont certains récursifs, dans cette syntaxe abstraite (vous pourrez éventuellement réaliser des extensions de syntaxe vous permettant d'avoir une syntaxe concrète).
2. Formaliser les règles d'évaluation en incluant les règles d'erreurs sous la forme d'une relation inductive. Écrire des lemmes correspondant à l'évaluation des exemples de programmes précédemment écrits. Écrire une tactique qui démontrent ces lemmes automatiquement (c'est-à-dire, une tactique qui réalise l'évaluation automatiquement).

$$\frac{n_{\mathbb{Z}} \in \mathbb{Z}}{\Gamma \vdash n \hookrightarrow n_{\mathbb{Z}}}\ \mathsf{int} \qquad \frac{}{\Gamma \vdash \mathsf{true} \hookrightarrow \top}\ \mathsf{true} \qquad \frac{}{\Gamma \vdash \mathsf{false} \hookrightarrow \bot}\ \mathsf{false} \qquad \frac{x \in \Gamma}{\Gamma \vdash x \hookrightarrow v}\ \mathsf{var}$$

$$\frac{\Gamma \vdash e \hookrightarrow n_{\mathbb{Z}}}{\Gamma \vdash\ -e \hookrightarrow\ -_{\mathbb{Z}}\ n_{\mathbb{Z}}}\ -_{\mathbb{Z}} \qquad \frac{\Gamma \vdash e_1 \hookrightarrow n_{1\mathbb{Z}} \quad \Gamma \vdash e_2 \hookrightarrow n_{2\mathbb{Z}}}{\Gamma \vdash e_1\ \mathsf{op}\ e_2 \hookrightarrow n_{1\mathbb{Z}}\ \mathsf{op}_{\mathbb{Z}}\ n_{2\mathbb{Z}}}\ \mathsf{op}_{\mathbb{Z}},\ \mathsf{op} \in \{+, -, \times\}$$

$$\frac{\Gamma \vdash e_1 \hookrightarrow n_{1\mathbb{Z}} \quad \Gamma \vdash e_2 \hookrightarrow n_{2\mathbb{Z}}}{\Gamma \vdash e_1\ \mathsf{op}\ e_2 \hookrightarrow n_{1\mathbb{Z}}\ \mathsf{op}_{\mathbb{Z}}\ n_{2\mathbb{Z}}}\ \mathsf{op}_{\mathbb{Z}},\ \mathsf{op} \in \{=, \neq, <, >, \leq, \geq\}$$

$$\frac{\Gamma \vdash e \hookrightarrow v_{\mathbb{B}}}{\Gamma \vdash \mathsf{not}(e) \hookrightarrow \mathsf{not}_{\mathbb{B}}(v_{\mathbb{B}})}\ \mathsf{not}_{\mathbb{B}} \qquad \frac{\Gamma \vdash e_1 \hookrightarrow v_{1\mathbb{B}} \quad \Gamma \vdash e_2 \hookrightarrow v_{2\mathbb{B}}}{\Gamma \vdash e_1\ \mathsf{op}\ e_2 \hookrightarrow v_{1\mathbb{B}}\ \mathsf{op}_{\mathbb{B}}\ v_{2\mathbb{B}}}\ \mathsf{op}_{\mathbb{Z}},\ \mathsf{op} \in \{\mathsf{and}, \mathsf{or}\}$$

$$\frac{\Gamma \vdash e_1 \hookrightarrow \top \quad \Gamma \vdash e_2 \hookrightarrow v_2}{\Gamma \vdash \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \hookrightarrow v_2}\ \mathsf{if}_{\top} \qquad \frac{\Gamma \vdash e_1 \hookrightarrow \bot \quad \Gamma \vdash e_3 \hookrightarrow v_3}{\Gamma \vdash \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \hookrightarrow v_3}\ \mathsf{if}_{\bot}$$

$$\frac{\Gamma \vdash e_1 \hookrightarrow v_1 \quad \Gamma, (x, v_1) \vdash e_2 \hookrightarrow v_2}{\Gamma \vdash \mathsf{let}\ x : \tau = e_1\ \mathsf{in}\ e_2 \hookrightarrow v_2}\ \mathsf{let}$$

$$\frac{}{\Gamma \vdash \mathsf{fun}\ (x\ :\ \tau) \to e \hookrightarrow <\Gamma, \mathsf{fun}\ (x\ :\ \tau) \to e>}\ \mathsf{fun}$$

$$\frac{}{\Gamma \vdash \mathsf{recfun}\ f\ (x\ :\ \tau_1)\ :\ \tau_2 \to e \hookrightarrow <\Gamma, \mathsf{recfun}\ f\ (x\ :\ \tau_1)\ :\ \tau_2 \to e>}\ \mathsf{recfun}$$

$$\frac{\begin{array}{c}\Gamma \vdash e_1 \hookrightarrow <\Gamma', \mathsf{fun}\ (x\ :\ \tau_1)\ :\ \tau_2 \to e_3> \\ e_2 \hookrightarrow v_2 \quad \Gamma', (x, v_2) \vdash e_3 \hookrightarrow v_3\end{array}}{\Gamma \vdash e_1\ e_2 \hookrightarrow v_3}\ \mathsf{app}_{\mathsf{fun}}$$

$$\frac{\begin{array}{c}\Gamma \vdash e_1 \hookrightarrow <\Gamma', \mathsf{recfun}\ f\ (x\ :\ \tau_1)\ :\ \tau_2 \to e_3> = c \\ e_2 \hookrightarrow v_2 \quad \Gamma', (f, c), (x, v_2) \vdash e_3 \hookrightarrow v_3\end{array}}{\Gamma \vdash e_1\ e_2 \hookrightarrow v_3}\ \mathsf{app}_{\mathsf{recfun}}$$

FIGURE 2 – Règles d'évaluation

3. Écrire une fonction qui réalise l'évaluation selon les règles d'évaluation en incluant également les règles d'erreurs. Tester cette fonction sur vos exemples de programmes. Démontrer que cette fonction est conforme à la relation inductive d'évaluation précédemment écrite.
4. Formaliser les règles de typage sous la forme d'une relation inductive. Écrire des lemmes correspondant au typage des exemples de programmes précédemment écrits. Écrire une tactique qui démontrent ces lemmes automatiquement (c'est-à-dire, une tactique qui réalise le typage automatiquement).
5. Écrire une fonction qui réalise le typage selon les règles de typage. Tester cette fonction sur vos exemples de programmes. Démontrer que cette fonction est conforme à la relation inductive de typage précédemment écrite.
6. Démontrer que le typage est correct vis-à-vis de la sémantique, c'est-à-dire que si un programme est bien typé et s'évalue sur une valeur, alors cette valeur n'est pas une erreur (notez que l'hypothèse « s'évalue sur une valeur » impose la terminaison du programme).

$$\frac{}{\Gamma \vdash n : \mathsf{int}}\ \mathsf{int} \qquad \frac{}{\Gamma \vdash \mathsf{true} : \mathsf{bool}}\ \mathsf{true} \qquad \frac{}{\Gamma \vdash \mathsf{false} : \mathsf{bool}}\ \mathsf{false} \qquad \frac{(x,\tau) \in \Gamma}{\Gamma \vdash x : \tau}\ \mathsf{var}$$

$$\frac{\Gamma \vdash e : \mathsf{int}}{\Gamma \vdash -e : \mathsf{int}}\ -_{\mathbb{Z}} \qquad \frac{\Gamma \vdash e_1 : \mathsf{int} \quad \Gamma \vdash e_2 : \mathsf{int}}{\Gamma \vdash e_1\ \mathsf{op}\ e_2 : \mathsf{int}}\ \mathsf{op}_{\mathbb{Z}},\ \mathsf{op} \in \{+,-,\times\}$$

$$\frac{\Gamma \vdash e_1 : \mathsf{int} \quad \Gamma \vdash e_2 : \mathsf{int}}{\Gamma \vdash e_1\ \mathsf{op}\ e_2 : \mathsf{bool}}\ \mathsf{op}_{\mathbb{Z}},\ \mathsf{op} \in \{=,\neq,<,>,\leq,\geq\}$$

$$\frac{\Gamma \vdash e : \mathsf{bool}}{\Gamma \vdash \mathsf{not}(e) : \mathsf{bool}}\ \mathsf{not}_{\mathbb{B}} \qquad \frac{\Gamma \vdash e_1 : \mathsf{bool} \quad \Gamma \vdash e_2 : \mathsf{bool}}{\Gamma \vdash e_1\ \mathsf{op}\ e_2 : \mathsf{bool}}\ \mathsf{op}_{\mathbb{Z}},\ \mathsf{op} \in \{\mathsf{and},\mathsf{or}\}$$

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 : \tau}\ \mathsf{if} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, (x,\tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \mathsf{let}\ x : \tau_1 = e_1\ \mathsf{in}\ e_2 : \tau_2}\ \mathsf{let}$$

$$\frac{\Gamma, (x,\tau_1) \vdash e : \tau_2}{\Gamma \vdash \mathsf{fun}\ (x\ :\ \tau_1) \to e : \tau_1 \to \tau_2}\ \mathsf{fun}$$

$$\frac{\Gamma, (f : \tau_1 \to \tau_2), (x,\tau_1) \vdash e : \tau_2}{\Gamma \vdash \mathsf{recfun}\ f\ (x\ :\ \tau_1)\ :\ \tau_2 \to e : \tau_1 \to \tau_2}\ \mathsf{recfun}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2}\ \mathsf{app}$$

FIGURE 3 – Règles de typage

3 Critères d'évaluation

Le projet doit être un fichier Coq, qui compile et qui réalise la tâche demandée. Mais il doit également être écrit de façon satisfaisante.

Le code du projet devra être bien présenté, au moyen d'une indentation correcte. Il devra être convenablement commenté. En particulier, il devra être indiqué clairement quels sont les paramètres et résultats de chaque fonction. De même, les preuves non triviales devront comporter des commentaires permettant au lecteur d'appréhender la structure de ces preuves. Des commentaires justifiant vos structures de données utilisées dans la formalisation seront également les bienvenus.

Le code devra être divisé en sous-fonctions et sous-lemmes de taille raisonnable (maximum une page d'écran). Il pourra y avoir plusieurs fichiers, mais ce n'est pas obligatoire. Le code devra être clair et concis.

4 Consignes

Le projet est individuel : chaque élève doit le réaliser et doit en écrire seul chacune des lignes de code. Vous avez le droit de vous entraider (c'est même conseillé), mais chacun d'entre vous travaille sur son code (ne vous échangez surtout pas de code).

Pendant la période de préparation du projet, vous pouvez demander l'aide à votre enseignant. Vous pouvez lui poser toutes les questions et lui soumettre tous vos problèmes. Sur le site du cours, il existe un forum, où vous pouvez poser également vos questions et échanger avec votre enseignant et les autres étudiants.

5 Remise du projet

Le projet terminé devra être rendu le 8 mai 2016 au plus tard. Il devra être rendu directement sur le site du cours à l'adresse (ne pas l'envoyer par mail à votre enseignant) :

`https://moodle.umontpellier.fr/course/view.php?id=1557`