Projet de DEA 2001/2002

Cours *Calcul des Constructions Inductives*

Présentation

L'objectif du projet est de formaliser le critère de bonne fondation de définitions par point-fixe pour les fonctions récursives.

Les projets sont individuels. Toute question est à adresser à l'une des personnes suivantes :

Bruno.Barras@inria.fr, Christine.Paulin@lri.fr, Hugo.Herbelin@inria.fr.

Fonctions récursives

Soit $\mathcal{R}$ le langage de termes suivant (x décrit un ensemble de variables de termes et f un ensemble de variables de fonctions)

$$t \quad ::= \quad \mathsf{O} \mid \mathsf{S}(t) \mid x \mid f(t) \mid \mathsf{match}\ t\ \mathsf{with}\ [\mathsf{O} \to t_1 | \mathsf{S}(x) \to t_2] \mid \mathsf{let\ rec}\ f\, x = t_1\ \mathsf{in}\ f(t_2)$$

Formaliser ce langage en Coq comme un type inductif de nom `terme`[1].

Le langage $\mathcal{R}$ permet de représenter les fonctions récursives : toute fonction récursive de n variables peut être représentée par un terme avec n variables libres (que l'on a ordonnées).

Par exemple, l'addition de x et y se représente par le terme

$$\text{“}x + y\text{”} \equiv \mathsf{let\ rec}\ f\, m = \mathsf{match}\ m\ \mathsf{with}\ [\mathsf{O} \to y | \mathsf{S}(m') \to \mathsf{S}(f(m'))]\ \mathsf{in}\ f(x)$$

où x et y sont des variables.

Alternativement, on peut représenter les fonctions récursives “à l'ordre supérieur” en utilisant les variables du méta-langage (appelées pour l'occasion méta-variables). Ainsi, la définition de l'addition devient

$$X + Y \equiv \mathsf{let\ rec}\ f\, m = \mathsf{match}\ m\ \mathsf{with}\ [\mathsf{O} \to Y | \mathsf{S}(m') \to \mathsf{S}(f(m'))]\ \mathsf{in}\ f(X)$$

où X et Y sont des méta-variables de termes, c'est-à-dire des variables Coq de type `term`. On retrouve alors la définition précédente en remplaçant X et Y par x et y respectivement.

Voici maintenant la fonction calculant la somme des entiers inférieurs à X

$$\mathsf{let\ rec}\ f\, n = \mathsf{match}\ n\ \mathsf{with}\ [\mathsf{O} \to \mathsf{O} | \mathsf{S}(n') \to \mathsf{S}(n' + f(n'))]\ \mathsf{in}\ f(X)$$

Définir la multiplication et la soustraction. Définir 3 fonctions, *paire*, $proj_1$ et $proj_2$, qui plongent bijectivement les paires d'entiers naturels dans les entiers naturels. On pourra avantageusement utiliser des méta-variables afin que les fonctions ainsi définies se comportent comme des macros :
(p.ex. `Definition mult [X,Y:terme] : terme := ...`).

La fonction d'Ackermann est définie par

$$\begin{array}{lcl} ack(0,n) & = & n+1 \\ ack(n+1,0) & = & ack(n,1) \\ ack(n+1,m+1) & = & ack(n, ack(n+1,m)) \end{array}$$

L'exprimer comme un terme du langage avec 2 variables libres.

[1] Attention, `Fix`, `Case`, `Cases` et `Match` sont des mots réservés de Coq !

Fonctions récursives primitives

Les fonctions récursives primitives sont les fonctions représentables par des termes (avec variables libres) de $\mathcal{R}$ pour lesquels les appels récursifs de fonctions définies par point-fixe se font sur des termes structurellement plus petits que l'argument initial du point-fixe.

Ordre de décroissance structurelle

On dit qu'un terme dans un type inductif est *clos* s'il n'est construit qu'à partir de constructeurs. Dans le cas du langage $\mathcal{R}$, les expressions ne dénotent que des entiers de Peano. Les termes clos de $\mathcal{R}$ sont donc les expressions construites à partir de O et S (c-à-d les entiers de Peano).

Un terme clos dans un type inductif est dit *structurellement plus petit* qu'un autre s'il s'obtient en retirant des constructeurs à cet autre terme. Dans le cas du langage $\mathcal{R}$, le seul constructeur non constant est S et on retombe donc sur l'ordre standard des entiers naturels.

L'ordre structurel sur les termes clos est décidable mais pas celui sur les termes arbitraires. Définir un prédicat `spp : termes -> termes -> Prop`, noté symboliquement $<_S$, qui capture au moins l'ordre structurel sur les entiers et idéalement, pour être utilisable en pratique, aussi l'ordre sur des termes avec variables (mais on pourra exclure les opérateurs de filtrage et de récursion).

Bonne fondation

On définit un prédicat $Prim(t)$ qui exprime que les appels récursifs d'un terme t sont tous bien fondés vis à vis de l'ordre structurel. Pour ce faire, on passe par un prédicat BF_σ paramétré par une liste σ de couples (f, t) avec f une fonction récursive et t un terme qui représente la forme structurée connue du terme initial de la récursion. Plus précisément, pour que $BF_\sigma(u)$ soit vérifié il est nécessaire que tous les appels à $f(v)$ dans u soient tels que v est plus petit que le terme t associé à f dans σ. Les règles qui définissent BF et $Prim$ sont les suivantes :

$$\frac{BF_\emptyset(t)}{Prim(t)} \qquad \frac{}{BF_\sigma(x)} \qquad \frac{}{BF_\sigma(\mathsf{O})} \qquad \frac{BF_\sigma(t)}{BF_\sigma(\mathsf{S}(t))} \qquad \frac{BF_\sigma(t) \qquad t \leq_S \sigma(f)}{BF_\sigma(f(t))}$$

$$\frac{BF_{\sigma[\mathsf{O}/x]}(t) \qquad BF_{\sigma[\mathsf{S}(y)/x]}(u)}{BF_\sigma(\mathsf{match}\ x\ \mathsf{with}\ [\mathsf{O} \to t | \mathsf{S}(y) \to u])} \qquad \frac{BF_{(\sigma, f \leftarrow x)}(t) \qquad BF_\sigma(u)}{BF_\sigma(\mathsf{let\ rec}\ f\, x = t\ \mathsf{in}\ f(u))}$$

où $\sigma(f)$ associe le terme initial de récursion avec appel récursif sur f et $\sigma[t/x]$ correspond à remplacer toute occurrence de x par t dans les termes figurant dans σ.

Par exemple, l'addition, définie ci-dessus, est bien fondée car, pour l'unique appel récursif, on a $m <_S \mathsf{S}(m)$ d'où $BF_{f \leftarrow \mathsf{S}(m)}(f(m))$ qui entraîne ultimement $Prim(x + y)$ en remontant la structure de l'expression représentant l'addition.

Formaliser le prédicat $Prim$ en Coq (`prim : terme -> Prop`).

Remarque : on pourra consulter les modules `Coq.Lists.PolyList` et `Coq.Lists.TheoryList` pour les fonctions standards sur les listes, ainsi que `Coq.Arith.Peano_dec` pour la décidabilité de l'égalité sur les entiers.

Exemple

Montrer que la définition de $x + y$ est bien fondée.

Test de bonne fondation

Écrire une fonction `teste_spp : terme -> terme -> bool` qui sait détecter l'ordre structurel au moins sur les termes pour lesquels on sait prouver l'ordre.

Montrer en Coq qu'effectivement cette fonction renvoie `true` dès que `spp` est prouvable.

Écrire une fonction `teste_bf : terme -> bool` qui implémente le prédicat de bonne fondation.

Prouver en Coq que cette fonction renvoie `true` pour les termes bien fondés.

Bonne fondation à réduction près

La définition du prédicat $Prim$ ci-dessus s'applique essentiellement à des expressions en forme normale vis à vis de la ι-réduction (et, pour bien faire, de la commutation des $\mathsf{match}\ t\ \mathsf{with}\ [\mathsf{O} \to t_1 | \mathsf{S}(x) \to t_2]$).

On aimerait pouvoir ajouter un cas de la forme suivante à la définition de BF :

$$\frac{BF_\sigma(u) \quad t \stackrel{h}{\to} u}{BF_\sigma(t)}$$

où $\stackrel{h}{\to}$ désigne la réduction faible de tête selon ι et les commutations de $\mathsf{match}\ t\ \mathsf{with}\ [\mathsf{O} \to t_1 | \mathsf{S}(x) \to t_2]$. Estimer quel pourrait-être le coût de cette généralisation. Pourra-t-on aussi garantir la terminaison de cette réduction ?

Autres extensions

La fonction

$$\mathsf{let\ rec}\ f\, n = \mathsf{match}\ n\ \mathsf{with}\ [\mathsf{O} \to \mathsf{match}\ n\ \mathsf{with}\ [\mathsf{O} \to \mathsf{O} | \mathsf{S}(m) \to f(\mathsf{S}(m))] | \mathsf{S}(p) \to \mathsf{O}]\ \mathsf{in}\ f(x)$$

est clairement bien fondée mais le critère échoue à le détecter. Pourrait-on étendre le critére de bonne fondation pour prendre en compte ce genre d'exemple limite ?

De même, la fonction

$$\mathsf{let\ rec}\ f\, m = \mathsf{match}\ (\mathsf{let\ rec}\ g\, n = \mathsf{match}\ n\ \mathsf{with}\ [\mathsf{O} \to \mathsf{O} | \mathsf{S}(p) \to g(p)]\ \mathsf{in}\ g(m))\ \mathsf{with}\ [\mathsf{O} \to \mathsf{O} | \mathsf{S}(p) \to f(p)]\ \mathsf{in}\ f(x)$$

est bien-fondée. Pourrait-on étendre le critére de bonne fondation pour le détecter ?

De même pour

$$\mathsf{let\ rec}\ f\, m = \mathsf{match}\ m\ \mathsf{with}\ [\mathsf{O} \to \mathsf{O} | \mathsf{S}(p) \to f(\mathsf{S}(p))]\ \mathsf{in}\ f(\mathsf{O})$$

Auriez-vous des raffinements à proposer ?

Fonctions récursives générales

Les fonctions récursives générales sont les fonctions totales représentables par des termes (avec variables libres) de $\mathcal{R}$ pour lesquels les appels récursifs de fonctions définies par point-fixe se font sur des termes prouvablement plus petits que l'argument initial du point-fixe.

À chaque fonction définie par point-fixe, on associe alors un ordre de décroissance sur les entiers. On fournit alors une preuve que chaque argument de chaque appel récursif décroît. Fournir une preuve sous-entend que l'on se place dans un formalisme logique $\mathcal{L}$ contenant $\mathcal{R}$ comme sous-langage. Notons $\vdash_{\mathcal{L}}$ la notion de dérivabilité pour cette logique.

La définition de bonne fondation est inchangée sauf dans les cas suivants.

$$\frac{BF_\emptyset(t)}{Rec_{\mathcal{L}}(t)} \qquad \frac{BF_\sigma(t) \qquad \sigma(f) = (R, u) \qquad \vdash_{\mathcal{L}} t\, R\, u}{BF_\sigma(f(t))}$$

$$\frac{\exists R \text{ relation sur les entiers de } \mathcal{L} \qquad BF_{(\sigma, f \leftarrow (R,x))}(t) \qquad BF_\sigma(u)}{BF_\sigma(\mathsf{let\ rec}\ f\,x = t\ \mathsf{in}\ f(u))}$$

Remarque : on retrouve notamment la version structurelle précédente (à la force logique de $\mathcal{L}$ près) en prenant la définition dans $\mathcal{L}$ de l'ordre strict usuel sur les entiers.

Formuler le théorème qui dit que si $\vdash_{\mathcal{L}}$ est cohérent, alors la bonne fondation d'un terme entraîne sa normalisation de tête (selon ι, la commutation du filtrage et le dépliage des points fixes) pour toute instantiation de ses variables libres.

On se dispensera de prouver ce théorème !

Modalités

Un rapport de quelques pages accompagné du source commenté du développement Coq est à rendre sous forme électronique pour le mardi 2 avril 12h, la soutenance aura lieu le mercredi 3 avril.