PFIOU: inférence de types pour ML

Yann Régis-Gianas

1 Objectifs du projet

Ce projet est l'application directe des deux premières séances du cours de typage des langages de programmation.

On étudie un langage de programmation fortement inspiré de ML. On vous fournit l'analyseur syntaxique ainsi qu'un ensemble de modules d'infrastructure (pretty-printer, syntaxe de contraintes, outils divers...).

L'objectif est d'implémenter :
- Un interpréteur pour ce langage ;
- La génération des contraintes de bon typage ;
- La résolution de ces contraintes.

La composition des deux derniers points conduit à l'obtention d'un moteur d'inférence de types.

2 Détails techniques

La tarball `pfiou-student.tar.gz` est un code à trou commenté. Le projet compile mais ne fonctionne évidemment pas : il reste à remplir les parties du projet contenant des `FIXME`.

Vous pouvez télécharger cette tarball à cette adresse :

`http://gallium.inria.fr/~regisgia/teaching/epita/pfiou`

Pour compiler le projet, il est nécessaire d'installer :
- O'Caml 3.09 (ou plus) : `http://caml.inria.fr` ;
- Menhir : `http://gallium.inria.fr/~fpottier/menhir` ;
- AlphaCaml : `http://gallium.inria.fr/~fpottier/alphaCaml`.

Une batterie de tests minimale est fournie dans la tarball (il est fortement conseillé de l'augmenter par vos propres tests).

Vous serez évalué lors d'une soutenance qui se déroulera courant septembre.

Voici les étapes du projet à réussir dans l'ordre (plus vous passez d'étapes et plus votre note sera haute) :

1. Interpréteur pour la totalité du langage ;
2. Inférence de type pour le sous-langage correspondant au λ-calcul simplement typé ;
3. Inférence de type pour le sous-langage correspondant à ML (l'ensemble du langage sans la récursion et les types algébriques) ;
4. Inférence de type en ajoutant la récursion ;
5. Inférence de type en ajoutant les types algébriques.

Les points suivants sont "bonus" (ils ne seront évalués qu'à condition que les étapes précédentes soient réussies) :
- Rajout de constantes au langage (par exemple les opérateurs de l'arithmétique, de comparaison, de manipulation de chaîne de caractère ou d'appel de fonctions externes) ;
- Ecriture de "gros programmes" à l'aide de ces constantes ;
- Rajout des types equi-récursifs ;
- Rajout des rangées ;
- Tout rajout intéréssant pourra être présenté.

3 Conseils

Avant toute chose, une lecture complète du code fourni s'impose pour comprendre l'intéraction entre les modules. La documentation d'AlphaCaml est nécessaire pour comprendre comment manipuler la syntaxe d'ordre supérieur. Vous pourrez ensuite attaquer l'interpréteur du langage. C'est une partie très simple. L'étape suivante est de commencer le solveur de contraintes (vous pouvez laisser de côté la généralisation pour le moment). En remplissant le module de génération de contraintes pour la restriction du langage au λ-calcul simplement typé. Reste alors à écrire l'algorithme de généralisation (la partie la plus difficile du projet) pour pouvoir traiter l'ensemble du langage de programmation excepté la récursion et les types algébriques. Ces deux derniers points sont relativement simples dans la mesure où il suffit de comprendre les contraintes associées à ces deux nouvelles constructions.