Simplification de contraintes de types

Julien Cohen

LaMI[1], Équipe SPÉCIF, UMR 8042 CNRS, Université d'Évry val d'Essonne, GENOPOLE
Tour Évry-2, 523 Place des terrasses de l'agora, 91000 Évry Cedex

2 décembre 2003

Nombre d'étudiants : 1 binôme

Mots-clés : sous-typage, inférence de types, types unions, collections hétérogènes.
Public visé : TER de maîtrise, stage IIE, stage Polytechnique.

Contexte de l'étude

Le projet `MGS` développe un langage de programmation original dédié à la modélisation et la simulation de processus biologiques à structure dynamique. Pour ce faire, `MGS` permet la représentation d'organisations complexes entre des entités variables et hétérogènes, ainsi que leur transformation par des règles locales. Ces travaux trouvent leurs inspirations dans les travaux de J. Von Neuman sur les automates cellulaires, A. Lindenmayer sur les L systèmes, G. Paun sur les P systèmes, G. Berry *et al.* sur la CHAM et la réécriture de multi-ensembles.

La structure de données fondamentale en `MGS` est la *collection topologique*. Une collection topologique est un ensemble d'éléments organisés par une relation de voisinage. Une *transformation* permet de spécifier de nouvelles fonctions sur les collections par des cas filtrant des *sous-collections*. Ces notions permettent d'unifier dans le même cadre formel les différents modèles de calculs cités plus haut. Pour chacun des modèles il suffit de choisir le bon voisinage pour la collection utilisée. Un point remarquable est l'existence d'un langage de filtres, utilisé pour écrire les règles d'une transformation, qui est commun à tous les types de collection. Ce langage de filtres se fonde sur la notion de voisinage et de chemin.

Sujet du stage

Le langage `MGS` peut être doté d'un système de types avec sous-typage. L'inférence de types en présence de sous-typage se décompose généralement en deux phases :

1. parcours du programe et génération de contraintes de types de la forme $\tau_1 < \tau_2$ où $<$ est la relation de sous-typage ;
2. résolution de l'ensemble des contraintes pour savoir si le programme admet un type.

Une erreur de type dans un programme se traduit par un ensemble de contraintes sans solution. Bien souvent, annoncer au programmeur que son programme est mal typé ne l'aide pas à trouver son erreur. Le fait que l'ensemble des contraintes produites est potentiellement illisible, car trop complexe, participe à la confusion du programmeur. C'est pourquoi des techniques ont été développées visant à simplifier un ensemble de contraintes pour rendre cet ensemble compréhensible par l'utilisateur[2].

Le but de ce stage est d'étudier les travaux de François Pottier et de proposer des techniques similaires adaptées au système de types de `MGS`. Ces propositions devront êtres implantées dans le moteur d'inférence de types de `MGS`.

Si la durée du stage le permet on pourra s'intéresser à d'autres problèmes dans le cadre de l'inférence de types avec sous-typage.

---

[1] *Contacts :* par courier électronique : `jcohen @ReMoVeMeFIRST.lami.univ-evry.fr`. Des informations supplémentaires sont disponibles à partir de la page : `http://mgs.lami.univ-evry.fr`

[2] Voir les travaux de François Pottier et notamment sa thèse :
`http ://pauillac.inria.fr/~fpottier/biblio/pottier.html#pottier-phd-french-98`