Proposition de TER M1

Des *code trees* adaptés à l'unification en logique du premier ordre

Sujet

Les termes en logique du premier ordre constituent l'unité de représentation de base de l'information dans plusieurs disciplines de l'informatique telles que la déduction automatique, la réécriture de termes, l'informatique symbolique et la programmation logique et fonctionnelle. Les opérations les plus usuelles sur les termes sont l'unification, le filtrage et la subsomption. Ces opérations sont souvent effectuées sur des collections de termes. Par exemple, on peut vouloir rechercher tous les termes qui s'unifient avec $f(x)$ dans l'ensemble $\{f(a), g(a, x), f(b)\}$, ce qui nous retournera $f(a)$ et $f(b)$.

Cette recherche de termes candidats qui ont une relation spécifique avec un terme de recherche donné est une opération centrale dans les outils de déduction automatique, les bases de données déductives et les systèmes de programmation logique et fonctionnelle. En l'absence de techniques permettant d'accélérer la recherche des termes candidats, le temps passé à identifier les candidats peut éclipser le temps passé à effectuer d'autres calculs utiles. Ce problème a suscité un grand intérêt de la part des chercheurs pour les techniques d'indexation des termes, c'est-à-dire les techniques de conception et de mise en œuvre de structures qui facilitent la recherche rapide d'un ensemble de termes candidats satisfaisant une certaine propriété à partir d'une grande collection de termes.

De manière générale, les techniques d'indexation reposent très largement sur des structures arborescentes et la recherche de termes candidats (par rapport à une requête donnée) revient à parcourir cet arbre. La structure à laquelle nous souhaitons nous intéresser ici est très différente. Ce sont les *code trees*, où la structure d'indexation est représentée cette fois-ci par un programme (un *code*) que l'on va exécuter sur la requête (qui sera vue comme son argument effectif).

Les *code trees* ont été introduits par Voronkov [2, 1] dans le cadre de l'implémentation de son outil de déduction automatique Vampire. Cet outil est basé sur la superposition et

manipule essentiellement des clauses. Voronkov a introduit les *code trees* pour gérer efficacement la subsomption, c'est-à-dire voir très rapidement quelles clauses d'un ensemble de clauses étaient subsumées par une clause donnée. Lorsque les clauses sont unitaires (un seul littéral), cela revient à un problème filtrage (*pattern matching*), à savoir que c_1 subsume c_2 s'il existe une substitution σ telle que $c_1\sigma = c_2$.

Dans ce TER, nous souhaitons généraliser les *code trees* de manière à pouvoir les utiliser pour faire de l'unification. Étant donné un terme t, le problème est donc de trouver tous les termes t' dans la structure d'indexation tels qu'il existe une substitution σ telle que $t\sigma = t'\sigma$. Ce travail va nécessiter de modifier la machine abstraite à subsomption de Voronkov, soit en rajoutant des instructions, soit en modifiant la sémantique des instructions existantes. Les *code trees* ont également une structure arborescente, on pourra de fait essayer de donner une sémantique concurrente à la machine obtenue. Enfin, on se propose d'implémenter la machine en question (dans un langage à déterminer) afin de voir si la structure obtenue est effective.

Travail à réaliser

— Modifier la machine abstraite de Voronkov pour traiter l'unification ;
— Donner une version séquentielle, puis concurrente de la sémantique ;
— Implémenter la machine obtenue (en version séquentielle ou concurrente).

Prérequis

— Avoir suivi un cours de logique de premier ordre.

Remarques additionnelles

L'encadrement du TER sera réalisé par :

— David Delahaye (Université de Montpellier, LIRMM, David.Delahaye@lirmm.fr) ;
— Hinde Bouziane (Université de Montpellier, LIRMM, Hinde.Bouziane@lirmm.fr) ;
— Julie Cailler (Université de Montpellier, LIRMM, Julie.Cailler@lirmm.fr).

Références

[1] I. V. Ramakrishnan, R. C. Sekar, and A. Voronkov. Term Indexing. In *Handbook of Automated Reasoning*, pages 1853–1964. Elsevier and MIT Press, 2001.

[2] A. Voronkov. The Anatomy of Vampire : Implementing Bottom-up Procedures with Code Trees. *Journal of Automated Reasoning (JAR)*, 15(2) :237–265, Oct. 1995.