

Logique de séparation et vérification déductive

Cette proposition de thèse se place dans le contexte de la plateforme Why [2], un ensemble d'outils pour la vérification déductive de programmes C et Java. À partir d'un programme source annoté par des spécifications formelles, la plateforme Why produit un ensemble de formules logiques exprimant la sûreté et la correction de ce programme, puis les transmet à des outils de preuve existants. La technologie employée consiste à combiner un calcul classique de plus faibles préconditions avec un modèle mémoire spécifique à chaque programme. Ce modèle mémoire exploite en premier lieu la séparation des différents champs composant les structures de données (champs de structures pour C et de classes pour Java) [1] puis une analyse statique de régions qui vient raffiner ce premier modèle [3]. Cette séparation statique de la mémoire permet de grandement simplifier les obligations de preuve.

Parallèlement à cette approche, la logique de séparation proposée par Reynolds et O'Hearn [5, 4] offre une alternative où l'utilisateur exprime par une conjonction séparante $P \star Q$ que les prédicats P et Q portent sur des portions disjointes de la mémoire. La logique de séparation permet donc d'exprimer des propriétés subtiles telles que « les deux listes chaînées p et q n'ont pas d'éléments en commun ». Un modèle mémoire basé sur une analyse statique tel que celui utilisé dans la plateforme Why ne pourra jamais offrir une telle propriété et ne permettra donc pas un raisonnement séparé sur les listes p et q . En contrepartie, la logique de séparation nécessite une logique de programmes propre et ne peut pas non plus exploiter les prouveurs automatiques existants.

L'objectif de cette thèse est de combiner ces deux approches de la preuve de programmes, pour tirer le meilleur parti de chacune. Plus précisément, cela signifie que l'utilisateur peut utiliser la conjonction séparante, mais continue de bénéficier des séparations déterminées de manière statique. Plusieurs approches sont possibles. Une première approche, naïve, peut consister à traduire la conjonction séparante dans une axiomatique du premier ordre, et à se ramener ainsi à la technologie actuelle de la plateforme Why. Cela implique notamment d'axiomatiser la notion de « portion de mémoire » (*footprint*), qui s'apparente à une notion d'ensembles finis, et de comprendre comment assurer un maximum de preuve automatique. Une autre approche, plus ambitieuse, consiste à modifier le calcul de plus faibles préconditions pour prendre en compte la conjonction séparante, tout en gardant les bénéfices du modèle mémoire statique actuel.

Une application immédiate de ce travail sera la simplification de la spécification formelle des programmes manipulant des structures chaînées (listes, arbres, graphes acycliques), par exemple dans des langages impératifs tels que C ou Java. Une contribution de la thèse devrait donc être l'extension des langages existants de spécification fonctionnelle par des notions nouvelles s'inspirant de la logique de séparation.

Encadrant : Jean-Christophe Filliâtre

Candidat potentiel : François Bobot

Références

- [1] Jean-Christophe Filliâtre and Claude Marché. Multi-prover verification of C programs. In Jim Davies, Wolfram Schulte, and Mike Barnett, editors, *6th International Conference on Formal Engineering Methods*, volume 3308 of *Lecture Notes in Computer Science*, pages 15–29, Seattle, WA, USA, November 2004. Springer.
- [2] Jean-Christophe Filliâtre and Claude Marché. The Why/Krakatoa/Caduceus platform for deductive program verification. In Werner Damm and Holger Hermanns, editors, *19th International Conference on Computer Aided Verification*, volume 4590 of *Lecture Notes in Computer Science*, pages 173–177, Berlin, Germany, July 2007. Springer.
- [3] Thierry Hubert and Claude Marché. Separation analysis for deductive verification. In *Heap Analysis and Verification (HAV'07)*, pages 81–93, Braga, Portugal, March 2007. <http://www.lri.fr/~marche/hubert07hav.pdf>.
- [4] Peter W. O'Hearn, John C. Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. In *CSL '01 : Proceedings of the 15th International Workshop on Computer Science Logic*, pages 1–19, London, UK, 2001. Springer-Verlag.
- [5] J. C. Reynolds. Separation logic : a logic for shared mutable data structures. In *17th Annual IEEE Symposium on Logic in Computer Science*. IEEE Comp. Soc. Press, 2002.