Vérification formelle d'une implémentation d'un gestionnaire de mémoire pour un compilateur certifié

Tahina Ramananandro
sous la direction de Xavier Leroy
Projet Gallium, INRIA Rocquencourt (France)

19 février – 26 juillet 2007

Le contexte général

Il s'agit de prouver formellement sur machine, en utilisant l'assistant de preuve Coq, une implémentation d'un gestionnaire de mémoire destinée à être intégrée à un compilateur certifié en cours de développement par l'équipe de recherche.

Le compilateur CompCert est un compilateur optimisant certifié de C vers le langage assembleur PowerPC via le langage intermédiaire Cminor. Il est en grande partie développé directement avec l'assistant de preuve Coq. Il est obtenu par l'extraction des preuves vers un langage de haut niveau, MiniML, un sous-ensemble d'OCaml. Le processus d'extraction a été certifié. Or, ce langage MiniML va aussi être compilé vers l'assembleur PowerPC en utilisant CompCert. On peut passer par Cminor, et utiliser CompCert pour la partie de Cminor vers PowerPC. On veut donc certifier le compilateur de MiniML vers Cminor. Celui-ci doit donc intégrer un gestionnaire de mémoire, écrit en Cminor et ajouté au code produit par la transformation, qu'il faut donc certifier.

Des GC ont déjà été prouvés formellement sur machine, notamment Yale : Andrew McCreight and Zhong Shao and Chunxiao Lin and Long Li, *A general framework for certifying garbage collectors and their mutators*, PLDI '07 : Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation, 2007. Il s'agit de la preuve d'un GC écrit directement en langage assembleur.

Le problème étudié

Nous avons tenté de prouver l'implémentation d'un GC (garbage collector).

En effet, le GC est une partie importante du gestionnaire de mémoire dans un compilateur certifié d'un langage de haut niveau. De nombreux GC sont mal écrits et bogués. Il est difficile de certifier un GC car celui-ci peut modifier profondément la structure de la mémoire.

La preuve de GC en elle-même n'est pas un problème nouveau, ni son interfaçage avec le gestionnaire de mémoire. Mais le fait de l'intégrer à un compilateur certifié est nouveau. En effet, CompCert est le premier compilateur certifié développé directement avec un assistant de preuve.

La contribution proposée

Nous avons tenté de prouver un GC mark and sweep avec coalescence (fusion des objets libres adjacents dans le tas). Pour cela, il fallait modéliser la structure de tas et de racines de la mémoire, et trouver les bons invariants afin de montrer que cette structure est préservée par les algorithmes.

Les arguments en faveur de sa validité

Cette solution est ouverte en ce sens qu'elle ne dépend pas du langage de haut niveau qu'on veut compiler.

Le problème, c'est que la preuve est extrêmement lourde avec l'assistant de preuve Coq. Beaucoup de problèmes techniques liés à l'implémentation actuelle de l'assistant de preuve imposent d'adapter les spécifications et les preuves. Il se peut que nous n'ayons pas les bons outils pour prouver in GC. Or, il faut pourtant pouvoir intégrer la preuve au compilateur certifié qui a été développé en utilisant Coq.

Le bilan et les perspectives

La modélisation du tas et des racines proposée ne dépend pas du GC. Elle offre un framework qui peut servir de base à la preuve d'autres GC, voire d'autres gestionnaires de mémoire pour d'autres langages de programmation de haut niveau qu'on voudrait compiler.

La preuve du GC, lorsqu'elle sera achevée et finalisée, sera intégrée au compilateur CompCert afin que la preuve de certification du compilateur de MiniML ou d'autres langages de haut niveau, vers Cminor, l'utilise.

Il faut achever la preuve, qui en est encore au stade du marquage car la preuve est extrêmement lourde.

Enfin, il faut se poser la question de l'interfaçage de la preuve du GC avec celle du gestionnaire de mémoire, notamment du mutateur.