Un analyseur flots de données paramétré par un ordre d'itération pour le compilateur formellement vérifié CompCert

Encadrant : Sandrine Blazy, professeur Univ Rennes - UMR IRISA

Dates : du lundi 14 mars au vendredi 29 juillet 2022

Description

Un compilateur vérifié est un logiciel accompagné d'une preuve de correction que le compilateur n'introduit pas de bogue dans les programmes. Une référence du domaine est le compilateur CompCert du langage C [1]. CompCert (comme sa variante CompCertSSA [2]) est un logiciel écrit dans le style purement fonctionnel de Coq, pour pouvoir ensuite être automatiquement extrait par Coq en du code OCaml qui peut être exécuté en dehors de Coq. Ceci permet entre autres de comparer les performances du compilateur avec celles d'un compilateur non vérifié, mais de l'état de l'art.

Plusieurs optimisations de CompCert utilisent une analyse flot de données générique [3], qui itère la résolution d'un ensemble d'équations selon un algorithme dû à Killdall [4]. L'objectif du stage sera de programmer et vérifier en Coq une nouvelle analyse flot de données paramétrée par un ordre d'itération du solveur d'équations, et de la valider sur une optimisation à ajouter dans CompCert.

L'ordre d'itération pourra résulter soit d'un parcours dans un arbre, soit d'un ensemble de noeuds à traiter [5-6]. Le solveur existant dans CompCert utilise un ordre fixe qui n'est pas adapté à des optimisations qui pourraient être ajoutées à CompCert. Par exemple, pour être efficaces, certaines optimisations nécessitent d'itérer sur un arbre de dominants, que CompCertSSA sait calculer. La technique de validation de l'analyse sera la validation *a posteriori*, et le validateur à programmer et vérifier en Coq s'inspirera de validateurs développés pour d'autres analyses [7]. Enfin, le stage comprendra une validation expérimentale mesurant précisément les performances des analyses comparées sur des exemples représentatifs, mettant en valeur les avantages et incovénients de ces différentes analyses.

Le stage comprend donc une partie de formalisation en Coq (et preuve de correction des ananlyses formalisées), ainsi qu'une partie de validation expérimentale du code OCaml extrait de la formalisation.

Bibliographie

[1] Formal verification of a realistic compiler. X.Leroy. Communications of the ACM, 52(7):107-115, 2009.

[2] A formally verified SSA-based middle-end - static single assignment meets CompCert. G.Barthe, D.Demange, and D.Pichardie. ESOP, 47-66, 2012

[3] Leroy, X. A Formally Verified Compiler Back-end. J Autom Reasoning 43, 363, 2009.

[4] A unified approach to global program optimization. G.Kildall. POPL, 194-206, 1973.

[5] Efficient chaotic iteration strategies with widenings. F.Bourdoncle. Formal Methods in Programming and Their Applications. LNCS 735, 1993.

[6] An empirical study of iterative data-flow analysis. K.Cooper, T. Harvey, K.Kennedy et al. IEEE conference on computing, 2006.

[7] S.Blazy, V.Laporte, A.Maroneze, D.Pichardie. Formal verification of a C Value Analysis Based on Abstract Interpretation. SAS 2013.

Modules associés

- 2-04 Programmation fonctionnelle et systèmes de types
- 2-06 Interprétation abstraite: application à la vérification et à l'analyse statique
- 2-07-1 Fondements des systemes de preuves
- 2-07-2 Assistants de preuves
- 2-36-1 Preuves de programmes