RPP : Preuve automatique de propriétés relationnelles par Self-Composition

Lionel Blatter, Nikolai Kosmatov, Pascale Le Gall, Virgile Prévosto

RPP : Preuve automatique de propriétés relationnelles par Self-Composition *

Lionel Blatter[1], Nikolai Kosmatov[1], Pascale Le Gall[2] and Virgile Prevosto[1]

[1] CEA, LIST, Software Reliability and Security Laboratory, 91191 Gif-sur-Yvette France
firstname.lastname@cea.fr
[2]Laboratoire de Mathématiques et Informatique pour la Complexité et les Systèmes
CentraleSupelec, Université Paris-Saclay, 91190 Gif-sur-Yvette France
firstname.lastname@centralesupelec.fr

Contexte. Les méthodes de vérification déductive basées sur la logique de Hoare [7] fournissent une approche puissante pour prouver qu'une fonction respecte certaines propriétés. Elles sont généralement couplées à un langage permettant de spécifier formellement les propriétés attendues, en particulier sous forme de contrats de fonction. Un tel contrat comprend des pré- et des post-conditions décrivant respectivement les entrées attendues et le comportement exigé de la fonction.

Problème. Cependant, toutes les propriétés qu'on peut vouloir établir sur un programme donné ne s'expriment pas facilement sous cette forme. En effet, un contrat de fonction décrit le déroulement d'une exécution d'une fonction donnée. Cependant, il est fréquent qu'on veuille parler d'une propriété relationnelle mettant en jeu l'exécution de plusieurs fonctions, ou comparer les résultats d'une même fonction sur différents paramètres. En particulier, des groupes de fonctions sont fréquemment liés par des spécifications algébriques [8] précisant leurs relations. Les contrats de fonction et les méthodes déductives classiques se prêtent mal à la spécification et à la vérification de telles propriétés. Des exemples de telles propriétés incluent :

$$\forall x, y; x <= y => f(x) <= f(y)$$

$$Decrypt(Crypt(Msg, PrivKey), PubKey) = Msg$$

On retrouve un sous ensemble des propriétés relationnelles dans le domaine du test, appelées *propriétés métamorphique* [6], liant plusieurs appels de la même fonction.

Contributions. Afin de répondre à ce manque de support pour les propriétés relationnelles lors de la vérification à base de contrats, nous proposons le plugin FRAMA-C/RPP présenté dans [4]. L'outil réalise un ensemble de transformations automatiques à partir d'un langage de spécification capable d'exprimer des propriétés relationnelles. Le résultat des transformations permet de prouver des propriétés relationnelles et de les utiliser comme hypothèse dans d'autres preuves en utilisant des outils de vérification déductive standard.

*Cet article est un résumé étendu de l'article [4] paru à TACAS 2017.

Approche. Notre langage de spécification est basé sur une extension du langage de spécification ACSL [3] afin de pouvoir exprimer ces propriétés relationnelles et ainsi pouvoir implémenter notre méthode de vérification dans un plugin FRAMA-C. Pour la vérification de propriétés relationnelles, RPP construit une fonction enveloppe, inspirée de la méthode de *Self-composition* [1], qui effectue les appels mis en jeu par la propriété dans un contexte adéquat. On est alors ramené à l'utilisation d'un calcul de plus faible précondition classique, effectué dans notre cas par le greffon FRAMA-C/WP [2].

En outre, afin de pouvoir utiliser les propriétés relationnelles, nous proposons une réécriture des propriétés sous forme axiomatique. WP ajoutera ces définitions dans le contexte des obligations de preuves suivantes.

Dans sa version actuelle [5], RPP supporte des fonctions avec effet de bord et des fonctions récursives. L'outil permet de traiter un large ensemble de propriétés relationnelles de manière automatique et de réutiliser les propriétés dans le contexte d'autres preuves. Il a également été testé avec succès sur différents *benchmarks*, ce qui a permis de valider notre approche.

Travaux futurs. Différents axes de recherches restent à explorer. Premièrement, la gestion des invariants de boucle sous forme d'invariant relationnel permettra de vérifier plus facilement des propriétés sur des fonctions avec boucles. Par ailleurs, la spécification de propriétés sur des appels de fonctions effectués le long d'une même trace d'exécution reste difficile, et pourrait être améliorée. Enfin, une utilisation directe du plugin WP par RPP, sans passer par une transformation syntaxique de code, permettra des gains de performance et une simplification de l'approche.

Références

[1] Barthe, G., D'Argenio, P.R., Rezk, T. : Secure information flow by self-composition. J. Mathematical Structures in Computer Science 21(6), 1207–1252 (2011)

[2] Baudin, P., Bobot, F., Correnson, L., Dargaye, Z. : WP Plugin Manual v1.0 (2017), `http://frama-c.com/download/frama-c-wp-manual.pdf`

[3] Baudin, P., Cuoq, P., Filliâtre, J.C., Marché, C., Monate, B., Moy, Y., Prevosto, V. : ACSL : ANSI/ISO C Specification Language, `http://frama-c.com/acsl.html`

[4] Blatter, L., Kosmatov, N., Gall, P.L., Prevosto, V. : RPP : automatic proof of relational properties by self-composition. In : Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2017). vol. 10205, pp. 391–397. Springer (2017)

[5] Blatter, L., Kosmatov, N., Le Gall, P., Prevosto, V., Petiot, G. : Static and Dynamic Verification of Relational Properties on Self-Composed C Code. In : Proceedings of the 12th Conference on Tests and Proofs (TAP 2018). To appear.

[6] Gotlieb, A., Botella, B. : Automated metamorphic testing. In : Proceedings of the 27th International Computer Software and Applications Conference (COMPSAC 2003). pp. 34–40. IEEE

[7] Hoare, C.A.R. : An axiomatic basis for computer programming. Communications of the ACM 12(10) (1969)

[8] Sannella, D., Tarlecki, A. : Foundations of Algebraic Specification and Formal Software Development. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2012)