Vérification de programmes avec pointeurs à l'aide de régions et de permissions

Romain Bardou

Résumé

La vérification déductive de programmes consiste à annoter des programmes par une spécification, c'est-à-dire un ensemble de formules logiques décrivant le comportement du programme, et à prouver que les programmes vérifient bien leur spécification. Des outils tels que la plate-forme Why prennent en entrée un programme et sa spécification et calculent des formules logiques telles que, si elles sont prouvées, le programme vérifie sa spécification. Ces formules logiques peuvent être prouvées automatiquement ou à l'aide d'assistants de preuve.

Lorsqu'un programme est écrit dans un langage supportant les alias de pointeurs, c'est-à-dire si plusieurs variables peuvent désigner la même case mémoire, alors le raisonnement sur le programme devient particulièrement ardu. Il est nécessaire de spécifier quels pointeurs peuvent être égaux ou non. Les invariants des structures de données, en particulier, sont plus difficiles à vérifier.

Cette thèse propose un système de type permettant de structurer la mémoire de façon modulaire afin de contrôler les alias de pointeurs et les invariants de données. Il est basé sur les notions de région et de permission. Les programmes sont ensuite interprétés vers Why de telle façon que les pointeurs soient séparés au mieux, facilitant ainsi le raisonnement. Cette thèse propose aussi un mécanisme d'inférence permettant d'alléger le travail d'annotation des opérations de régions introduites par le langage. Un modèle est introduit pour décrire la sémantique du langage et prouver sa sûreté. En particulier, il est prouvé que si le type d'un pointeur affirme que celui-ci vérifie son invariant, alors cet invariant est effectivement vérifié dans le modèle. Cette thèse a fait l'objet d'une implémentation sous la forme d'un outil nommé Capucine. Plusieurs exemples ont été écrits pour illustrer le langage, et ont été vérifié à l'aide de Capucine.