Verification of Programs With Pointers Using Regions and Permissions

Romain Bardou

Abstract

Deductive verification consists in annotating programs by a specification, i.e. logic formulas which describe the behavior of the program, and prove that programs verify their specification. Tools such as the Why platform take a program and its specification as input and compute logic formulas such that, if they are valid, the program verifies its specification. These logic formulas can be proven automatically or using proof assistants.

When a program is written in a language supporting pointer aliasing, i.e. if several variables may denote the same memory cell, then reasoning about the program becomes particularly tricky. It is necessary to specify which pointers may or may not be equal. Invariants of data structures, in particular, are harder to maintain.

This thesis proposes a type system which allows to structure the heap in a modular fashion in order to control pointer aliases and data invariants. It is based on the notions of region and permission. Programs are then translated to Why such that pointers are separated as best as possible, to facilitate reasoning. This thesis also proposes an inference mechanism to alleviate the need to write region operations introduced by the language. A model is introduced to describe the semantics of the language and prove its safety. In particular, it is proven that if the type of a pointer tells that its invariant holds, then this invariant indeed holds in the model. This work has been implemented as a tool named Capucine. Several examples have been written to illustrate the language, and where verified using Capucine.