

OO Specification for Verification of OO Programs

(Abstract)

Qiu Zongyan and Yijing Liu

LMAM and Department of Informatics, School of Mathematical Sciences, Peking University

Email: {liuyijing, hongali, qzy}@math.pku.edu.cn

Specification and verification for object oriented (OO) programs remains a great challenge despite of decades' efforts. To address this problem, we propose a novel specification and verification framework, which supports abstraction and offers modularity via a concept called *specification predicate*. It covers many important OO features like encapsulation, information hiding, inheritance and polymorphism.

To specify and verify OO programs, we need to consider various issues:

- In OO practice, developers distinguish interface from implementation, to prevent close dependence on implementation details and achieve high degree of modularity. The case should be the same in formal specification and verification. For better modularity in formal work, we need to suppose specification on abstract level, which can be completely independent of the code if desired.
- Having abstract specifications, we need some way to link them with the code in the class. Clearly, the link may involve any implementation detail. However, as in programming practice, we may not want some details to leak out, thus need locality and visibility.
- Two issues above are general to programs with data abstraction. For OO programs, we have also to consider inheritance and overriding. Having a class well-specified and verified, and introduce its subclass, we want that the existing specification and verification can be reused.

Our framework embodies these considerations.

- We propose a framework which supports abstract level specification for information hiding and encapsulation. To support modular specification and verification, we integrate the specification facilities with important OO features, and define rules for inheritance, overriding, encapsulation, and visibility of specifications.
- We propose a concept *specification predicate* to connect abstract specification with implementation details. The predicates play a key role in modular verification of OO programs with information hiding and dynamic behavior.
- To present our ideas, a small OO language with specification features, VeriJ, is defined. By the language design and its embedded verification framework, we show how the encapsulation, information hiding, inheritance, polymorphism, etc. can enhance the formal verification ability for OO programs. We define a set of Hoare-style rules for generating proof obligations.
- We use some examples to illustrate how specification and verification can be carried out in our framework modularly, and how our approach can bridge the gaps between a verification logic and implementation details smoothly.