

Mechanizing Programming Logics in Higher Order Logic

Michael J.C. Gordon

Computer Laboratory
New Museums Site
Pembroke Street
Cambridge CB2 3QG

SRI International
Suite 23
Millers Yard
Cambridge CB2 1RQ

Abstract:

Formal reasoning about computer programs can be based directly on the semantics of the programming language, or done in a special purpose logic like Hoare logic. The advantage of the first approach is that it guarantees that the formal reasoning applies to the language being used (it is well known, for example, that Hoare's assignment axiom fails to hold for most programming languages). The advantage of the second approach is that the proofs can be more direct and natural.

In this paper, an attempt to get the advantages of both approaches is described. The rules of Hoare logic are *mechanically* derived from the semantics of a simple imperative programming language (using the HOL system). These rules form the basis for a simple program verifier in which verification conditions are generated by LCF-style tactics whose validations use the derived Hoare rules. Because Hoare logic is derived, rather than postulated, it is straightforward to mix semantic and axiomatic reasoning. It is also straightforward to combine the constructs of Hoare logic with other application-specific notations. This is briefly illustrated for various logical constructs, including termination statements, VDM-style 'relational' correctness specifications, weakest precondition statements and dynamic logic formulae .

The theory underlying the work presented here is well known. Our contribution is to propose a way of mechanizing this theory in a way that makes certain practical details work out smoothly.

Contents

1	Introduction	389
2	A Simple Imperative Programming Language	390
3	Hoare Logic	391
3.1	Axioms and Rules of Hoare Logic	392
3.2	An Example Proof in Hoare Logic	394
4	Higher Order Logic	396
4.1	Types	397
4.2	Definitions	399
5	Semantics in Logic	399
5.1	Semantics of Commands	401
5.2	Semantics of Partial Correctness Specifications	403
6	Hoare Logic as Higher Order Logic	403
6.1	Derivation of the <code>skip</code> -axiom	404
6.2	Derivation of Precondition Strengthening	405
6.3	Derivation of Postcondition Weakening	405
6.4	Derivation of the Sequencing Rule	406
6.5	Derivation of the <code>if</code> -rule	406
6.6	Derivation of the <code>while</code> -rule	406
6.7	Derivation of the Assignment Axiom	407
7	Hoare Logic in HOL	408
8	Introduction to Tactics and Tacticals	415
8.1	Tactics	415
8.2	Using Tactics to Prove Theorems	416
8.3	Tacticals	417
9	Verification Conditions via Tactics	417
10	Termination and Total Correctness	424
10.1	Derived Rules for Total Correctness	426
10.2	Tactics for Total Correctness	426
11	Other Programming Logic Constructs	428
11.1	VMD-Style Specifications	429
11.2	Dijkstra's Weakest Preconditions	431
11.3	Dynamic Logic	435
12	Conclusions and Future Work	436
13	Acknowledgements	437