Hoare Logic and Auxiliary Variables

Thomas Kleymann
Laboratory for Foundations of Computer Science, The University of Edinburgh, Scotland

Keywords: Hoare Logic; Auxiliary variables; Adaptation Completeness; Most General Formula; VDM

Abstract. Auxiliary variables are essential for specifying programs in Hoare Logic. They are required to relate the value of variables in different states. However, the axioms and rules of Hoare Logic turn a blind eye to the role of auxiliary variables. We stipulate a new structural rule for adjusting auxiliary variables when strengthening preconditions and weakening postconditions. Courtesy of this new rule, Hoare Logic is adaptation complete, which benefits software re-use. This property is responsible for a number of improvements. Relative completeness follows uniformly from the Most General Formula property. Moreover, one can show that Hoare Logic subsumes Vienna Development Method’s (VDM) operation decomposition rules in that every derivation in VDM can be naturally embedded in Hoare Logic. Furthermore, the new treatment leads to a significant simplification in the presentation for verification calculi dealing with more interesting features such as recursion.

1. Introduction

Hoare Logic is a verification calculus which relates imperative programs with two assertions, both (first-order) logical formulae. These assertions are interpreted as predicates on states where free variables denote the value of program variables in a specific state. Variables for which no counterpart appears as a program variable in the program under consideration then take on the role of auxiliary variables. They are required to relate the value of program variables in different states.

In practice, auxiliary variables are essential ingredients for specifying properties about imperative programs [Vic91]. Nevertheless, the axioms and rules in
Hyare Logic do not support the role of auxiliary variables. This is a known deficiency and has been overcome in other frameworks e.g., specification logic [Rey82] and the Vienna Development Method (VDM) [Jon90].

In our opinion, the role of auxiliary variables in Hoare Logic has been underestimated. We stipulate a new structural rule for adjusting auxiliary variables when strengthening preconditions and weakening postconditions. This alone leads to adaptation completeness. One may adapt arbitrary satisfiable specifications. As a consequence,

- we clarify how to uniformly establish completeness as a corollary of Gorelick’s Most General Formula (MGF) theorem [Gor75] which focusses on deriving a specific correctness formula. One may adapt the MGF specification to an arbitrary specification in a single step.
- We can show that Hoare Logic subsumes VDM’s operation decomposition rules in that every derivation in VDM can be naturally embedded in Hoare Logic.
- The Hoare Logic presentation for recursive procedures can be simplified significantly. Specifically, we are able to show that Sokołowski’s calculus [Sok77] is sound and complete if one replaces Hoare’s rule of consequence with ours. Apt’s remedy of adding three further structural rules had led to a complete but unsound system [Apt81, AdB90].

The overview of this paper is as follows: Hoare Logic and the concepts of soundness, relative completeness and adaptation completeness are briefly introduced in Section 2.

In Section 3, we discuss the role of auxiliary variables in Hoare Logic. As our main contribution, we stipulate an improved rule of consequence which allows us to modify auxiliary variables while strengthening preconditions and weakening postconditions. The section concludes with a proof of adaptation completeness.

In Sections 4–7, we illustrate the benefits of our new approach. In Section 4, we present a language independent completeness proof as a corollary of the MGF property. In Section 5, we establish that VDM’s operation decomposition rules can be simulated in Hoare Logic. In Section 6, we review the development of verification calculi for imperative programs with recursive (parameterless) procedures in the setting of total correctness. Our treatment of auxiliary variables leads to a significant simplification. We turn our attention to calculi for concurrent programs in Section 7. By way of examples, we attempt to point out weaknesses in the standard Hoare Logic rules.

2. Hoare Logic

For the purpose of this section, we consider a (very) simple imperative programming language merely consisting of assignments, sequential composition, conditionals and loops.

**Definition 2.1. (Syntax of Programs).** Imperative programs \( S : \text{prog} \) are defined by the BNF grammar

\[
S ::= x := e \mid S_1 ; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{while } b \text{ do } S
\]

where \( x \) is a program variable, \( e \) an expression and \( b \) a boolean expression.