Metatheory of Verification Calculi in LEGO*

To What Extent Does Syntax Matter?

Thomas Kleymann

LFCS Edinburgh
King’s Buildings, Mayfield Road, Edinburgh EH9 3JZ, Scotland

Abstract. Investigating soundness and completeness of verification calculi for imperative programming languages is a challenging task. Incorrect results have been published in the past. We take advantage of the computer-aided proof tool LEGO to interactively establish soundness and completeness of both Hoare Logic and the operation decomposition rules of the Vienna Development Method with respect to operational semantics. We deal with parameterless recursive procedures and local variables in the context of total correctness. In this paper, we discuss in detail the role of representations for expressions, assertions and verification calculi. To what extent is syntax relevant? One needs to carefully select an appropriate level of detail in the formalisation in order to achieve one’s objectives.

1 Introduction

We have taken advantage of the LEGO system [1] to produce machine-checked soundness and completeness proofs for Hoare Logic and the operation decomposition rules of the Vienna Development Method (VDM). Our imperative programming language includes (parameterless) recursive procedures and local variables. We consider static binding and total correctness. This is one of the largest developments in LEGO to date. Building on a comprehensive library it additionally consists of more than 800 definitions, lemmata and theorems.

Our message to the designers and researchers of verification calculi is that conducting computer-aided soundness and completeness proofs is both a feasible and profitable task. Our fundamental contribution has been to highlight the role of auxiliary variables in Hoare Logic. Usually, assertions are interpreted as predicates on states where free variables denote the value of program variables in a specific state. Variables which are unaffected by 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.

Our view of assertions emphasises the pragmatic importance of auxiliary variables. We have followed a proposal by Apt & Meertens to consider assertions as relations on states and auxiliary variables [2]. Furthermore, we stipulate a new structural rule to adjust auxiliary variables when strengthening preconditions.


© Springer-Verlag Berlin Heidelberg 1999
and weakening postconditions. This rule is stronger than all previously suggested structural rules, including Hoare’s consequence rule [3] and rules of adaptation. As a direct consequence of the new treatment of auxiliary variables,

- we were able to show that Sokolowski's calculus for recursive procedures [4] is sound and complete if one replaces Hoare’s rule of consequence with ours. In particular, none of the other structural rules introduced by Apt [5] (which lead to a complete but unsound system) are required.
- We have clarified the relationship between Hoare Logic and its variant VDM. We were able to show that, contrary to common belief, VDM is more restrictive than Hoare Logic in that every derivation in VDM can be naturally embedded in Hoare Logic.

**Deep versus Shallow Embedding.** Traditionally, one defines syntax for expressions and relative to this setup, one characterises syntax of a programming language and syntax of an assertion language. Then, one describes the meaning of every syntactic construct. This approach is known as **deep embedding**. Alternatively one may shortcut this process and identify the syntactic representation with its denotation. This technique is known as **shallow embedding**.

**Related Work.** The pioneering work on machine-checked soundness for Hoare Logic by Gordon [6] rests entirely on shallow embedding. Homeier [7] extends the soundness proof to a setting with mutually recursive procedures. His encoding is based exclusively on deep embedding. Nipkow [8] has been the first to conduct a machine-checked **completeness** proof for Hoare Logic dealing with simple imperative programs in the context of partial correctness. This contains a mixture of shallow and deep embedding. Using similar representation techniques we have extended this work to recursive procedures and local variables.

### 1.1 To What Extent Does Syntax Matter?

Before deciding on the embedding technique, one ought to clarify the objectives of the machine-assisted development. This induces the level of detail in which one needs to analyse involved concepts. One of the central issues in formalising metatheory is to what extent syntax needs to be formalised. Technically, one has a choice of deep versus shallow embedding.

A shallow embedding cuts down the work load and is therefore, at least for machine-checked developments, often the preferred approach. The drawbacks of shallow embedding are that

1. one cannot exploit the inductive (syntactic) structure to prove properties.
2. The representation of concrete examples is often more difficult to comprehend.

As the main contribution of this paper, we clarify the role of deep versus shallow embedding. In the setting of Hoare Logic, the choice of the level of embedding has a major influence in the work involved in setting up an appropriate