Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information and Computation 199 (2005) 200–227

Information and Computation

www.elsevier.com/locate/ic

Proving pointer programs in higher-order logic

Farhad Mehta[a,*,1], Tobias Nipkow[b]

[a]*Department of Computer Science, ETH Zürich, Switzerland*
[b]*Institut für Informatik, Technische Universität München, Germany*

Received 17 November 2003; revised 29 July 2004
Available online 28 January 2005

Abstract

Building on the work of Burstall, this paper develops sound modelling and reasoning methods for imperative programs with pointers: heaps are modelled as mappings from addresses to values, and pointer structures are mapped to higher-level data types for verification. The programming language is embedded in higher-order logic. Its Hoare logic is derived. The whole development is purely definitional and thus sound. Apart from some smaller examples, the viability of this approach is demonstrated with a non-trivial case study. We show the correctness of the Schorr–Waite graph marking algorithm and present part of its readable proof in Isabelle/HOL.
© 2005 Elsevier Inc. All rights reserved.

Keywords: Pointer programs; Verification; Hoare logic; Higher-order logic; Schorr–Waite algorithm

1. Introduction

It is a truth universally acknowledged, that the verification of pointer programs must be in want of machine support. The basic idea in all approaches to pointer program proofs is the same and

[*] Corresponding author. Fax: +41 44 632 1435.
E-mail address: fmehta@inf.ethz.ch (F. Mehta).
[1] Work done while at the Technische Universität München, supported by DFG Grant NI 491/6-1.

0890-5401/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2004.10.007

goes back to Burstall [6]: model the heap as a collection of variables of type *address* $\rightarrow$ *value* and reason about the programs in Hoare logic. A number of refinements of this idea have been proposed; see [16] for a partial bibliography. The most radical idea, again inspired by Burstall, is that of *separation logic* [17]. Although very promising, it is difficult to combine with existing theorem proving infrastructure because of its special logical connectives. Instead we take Bornat's [3] presentation of Burstall's ideas as our point of departure.

Systematic approaches to automatic or interactive verification of pointer programs come in two flavours. There is a large body of work on program analysis techniques for pointer programs. These are mainly designed for use in compilers and can only deal with special properties like aliasing. In the long run these approaches will play an important role in the verification of pointer programs. But we ignore them for now because our goal is a general purpose logic. For the same reason we do not discuss other special purpose logics, e.g. [9].

General theorem proving approaches to pointer programs are few. A landmark is the thesis by Suzuki [20] who developed an automatic verifier for pointer programs that could handle the Schorr–Waite algorithm. However, that verification is based on 5 recursively defined predicates (which are not shown to be consistent – mind the recursive "definition" $P = \neg P$!) and 50 unproved lemmas about those predicates. Bornat [3] has verified a number of pointer programs with the help of Jape [4]. However, his logical foundations are a bit shaky because he choses not to address definedness and soundness issues, leaving many lemmas unproven. Furthermore, since Jape is only a proof editor with little automation, the Schorr–Waite proof takes 152 pages [5]. Apart from the works of Suzuki and Bornat, we are unaware of any other mechanically checked proofs of the Schorr–Waite algorithm.

The contributions of our paper are as follows:

- An embedding of a Hoare logic for pointer programs in a general purpose theorem prover (Isabelle/HOL).
- A logically fully sound method for the verification of inductively defined data types like lists and trees on the pointer level.
- A treatment of cyclic pointer structures using similar methods.
- A readable and machine checked proof of the Schorr–Waite algorithm.

Instrumental in achieving the second point is the replacement of Bornat's recursive abstraction functions, which require a logic of partial functions, by inductive relations, which are definable in a logic of total functions. This is crucial, as most exiting theorem provers support total functions only.

The point about "readable" proofs deserves special discussion as it is likely to be controversial. Our aim was to produce a proof that is close to a journal-style informal proof, but written in a stylised proof language that can be machine-checked. Isabelle/Isar [21,11], like Mizar, provides such a language. Publishing this proof should be viewed as creating a reference point for further work in this area: although an informal proof is currently shorter and more readable, our aim should be to bridge this gap further. It also serves as a reference point for future mechanisations of other formal proofs like the separation logic one by Yang [22].

The rest of the paper is structured as follows. After a short overview of Isabelle/HOL (Section 2) and an embedding of a simple imperative programming language in Isabelle/HOL (Section 3), we