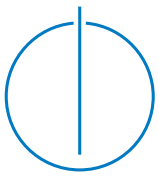
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Formalization of an Imperative Language with Procedures, Arrays and Pointers in Isabelle/HOL

Gabriela Limonta



DEPARTMENT OF INFORMATICS

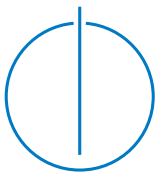
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Formalization of an Imperative Language with Procedures, Arrays and Pointers in Isabelle/HOL

Formalisierung einer Imperativen Sprache mit Prozeduren, Arrays und Zeigern in Isabelle/HOL

Author:	Gabriela Limonta
Supervisor:	Prof. Tobias Nipkow
Advisor:	Dr. Peter Lammich
Submission Date:	TODO: Submission date



I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, TODO: Submission date Gabriela Limonta

Acknowledgments

Abstract

Contents

1 Introduction

1.1 Motivation

1.2 Theoretical Background

1.2.1 Semantics of a programming language

Definition

Types of semantics

1.2.2 HOL

1.2.3 Language used in this work

The language should probably have a name. I have to give one to it.

2 Previous and Related Work

2.1 CompCert Project

2.1.1 Program Logics Andrew W. Appel

Relevant to this part is the Program Logics for Certified Compilers book from Andrew W. Appel. This project uses the same memory model used in this work and that's why it's relevant.

2.2 C formalized in HOL, Michael Norrish

Norrish has a formalized C semantics in HOL, our semantics is also formalized in HOL.

2.3 Different approach (From C code to semantics)

Mildly interesting, the AutoCorres project parses C code and abstracts it as many other VCGs out there. We are **generating** C code, this is the opposite direction.

3 Syntax and Semantics

3.1 Isabelle

Why is Isabelle chosen to use in this work instead of some other tool. IsabelleHOL, Isabelle's code generation. Maybe this should be moved to the theoretical background.

3.2 Expressions

3.2.1 Syntax

Expressions

Memory

Types

- Addresses
- Integers

3.2.2 Semantics

Here is where the semantics for the evaluation of expressions belongs.

3.3 Commands

3.3.1 Syntax

3.3.2 Semantics

Functions

Programs

3.4 Restrictions

Here we mention the architecture restrictions (64 bit architectures) this semantics assumes. Not sure if int width should be explained here or before in the integer types.

3.5 States

3.5.1 Valuation

3.5.2 Stack

3.5.3 Procedure Table

3.6 Small Step Semantics

3.6.1 CFG

3.6.2 Small Step semantics rules

3.7 Interpreter

3.7.1 Single step

3.7.2 Execution and Interpretation

3.7.3 Correctness

4 Pretty Printer

4.1 Values

4.2 Memory

4.3 Expressions

4.4 Commands

4.5 Declarations

4.6 States

4.7 Programs

4.8 Exporting C code

5 Testing

5.1 Test Harness in Isabelle

5.2 Test Harness in C

5.3 Tests

5.3.1 Equality of final states or configurations

Final state of reachable memory and global variables yielded by the Isabelle interpreter is the same as the one yielded by the generated, compiled and run C program.

5.3.2 Generation of Tests

Integer Values

Pointers

5.3.3 Generation of code with tests

using the test harness and the C macros. The semantics stays the same during the translation process. What properties are covered by this test suite?

5.4 Example programs

6 Results

7 Conclusion and Future Work

Future work includes:

- Parsing C tests from the gcc torture test suite and translating to our semantics in order to generate and test C code from them.
- Formalize proof rules for the language, separation logic for pointers
- Link the language with the Isabelle Refinement Framework so programs from the Isabelle Refinement Framework can be refined to programs in this language.
- Expand the set of expressions and instruccions, doesn't add computational power to the language but adds comfort for the programmer.
- Add a type system.

Glossary

computer is a machine that....

Acronyms

TUM Technische Universität München.

List of Figures

List of Tables