Proof in VDM: case studies

Edited by:
Juan Bicarregui

With contributions from:
Sten Agerholm
Bart Van Assche
John Fitzgerald
Jacob Frost
Albert Hoogewijs
Cliff Jones
Peter Lindsay
Savi Maharaj
Brian Matthews
Paul Mukherjee
Noemie Slaats

Preface

Not so many years ago, it would have been difficult to find more than a handful of examples of the use of formal methods in industry. Today however, the industrial application of formal methods is becoming increasingly common in a variety of application areas, particularly those with a safety, security or financially critical aspects. Furthermore, in situations where a particularly high level of assurance is required, formal proof is broadly accepted as being of value.

Perhaps the major benefit of formalisation is that it enables formal symbolic manipulation of elements of a design and hence can provide developers with a variety of analyses which facilitate the detection of faults. Proof is just one of these possible formal activities, others, such as test case generation and animation, have also been shown to be effective bug finders. Proof can be used for both validation and verification. Validation of a specification can be achieved by proving formal statements conjectured about the required behaviours of the system. Verification of the correctness of successive designs can be achieved by proof of a prescribed set of proof obligations generated from the specifications.

The development of accurate formal specifications is clearly a difficult, time consuming and error prone task. Constructing proofs - typically several times larger than the software they are concerned with - is no simpler, but its value comes not only from of the degree of assurance given by proof, but also from the process of proof itself. Often, attempting a proof, that may well be impossible because of some mistake in the informal reasoning that led to the design, will uncover the precise nature of the fault and indicate the required correction. Other times, even though a design may be correct, the proof activity may lead us to a clearer, more elegant specification, which will in turn lead to higher quality software.

The automatic checking of the correctness of a completed fully formal proof is a relatively straightforward task which can be performed by a simple program that it is possible to trust. However, the construction of such proofs is at best computationally complex and, in general, impossible to automate. In some circumstances, it may be possible to lessen the cost of proof development by allowing rigorous rather than fully formal proof, but this increases the complexity of proof checking and hence reintroduces the possibility of acceptance of an incorrect proof. Unlike mathematical proofs which are repeatedly studied by successive generations of scholars, proofs of software are unlikely to be of interest outside the development team, unless such inspections are built into the development process.

This book describes a collection of case studies in the use of formal and rigorous proof in the validation and verification formal specifications mostly in safety- or security-critical application areas. In an earlier book in this series, "Proof in VDM: A Practitioner's Guide", the editor and several contributors explained the basic principles of proof and illustrated how it can be used in practice in VDM. This book is in many ways a companion to that text, presenting the use those techniques in a range of actual applications. Again, attention is focused on VDM-SL because of its ISO standardisation and because it has a well-documented and well-developed proof theory.

The case studies illustrate different aspects of the use of proof in formal development covering the validation of security and safety properties and the verification of formal refinement. The first four chapters describe case studies in which the proofs were developed by hand. These proofs are presented in considerable detail although not fully formally. The remaining chapters deal with machine supported, fully formal proof, employing a variety of support tools. The tools described are Mural, PVS and Isabelle.

The first case study describes *a formal model of a nuclear process tracking system* which was developed for the UK Health and Safety Executive. It presents the formal model itself, the elicitation and formalisation of safety properties from the model, and the proof that the model is consistent with respect to these properties. The study illustrates issues in the modular structuring of specifications and proofs as well as questions of methodology and specification style.

The second case study describes *the validation of an explosives store* against informally expressed requirements. This chapter uses an existing specification of UN regulations for safe storage of explosives to illustrate and compare a range of validation techniques available to the developer. Issues considered include levels of type checking, testing of executable specifications, and proof.

The third chapter describes *the specification and validation of a security policy model* for a network handling sensitive and classified material. The security policy model is specified and validated by showing that it is mathematically consistent and satisfies certain security properties. Some new techniques concerning proof obligations for exception conditions in VDM-SL are described.

The fourth chapter describes *the specification and proof of an EXPRESS to SQL compiler*. An abstract specification of an EXPRESS database is given and then refined by an implementation on top of a SQL relational database. The compiler is thus formalised as a refinement and the equivalence of the abstract and concrete specifications becomes the justification of its correctness.

The fifth chapter presents *a unified formalisation of shared memory models* for explicitly parallel programs, giving the semantics of the memory access and synchronisation instructions. It describes how the Mural tool was used in writing the VDM specification, generating the corresponding formal theory and constructing some fully formal proofs of basic properties of the model.

The sixth chapter describes *the use of the PVS system to support proof in VDM*.

It presents an easy and direct translation from VDM-SL into the PVS specification language and describes the use of PVS for typechecking and verifying properties of VDM-SL specifications and refinements. The translation and possibilities for proof arising from it are illustrated through a number of small examples and one larger case study of a protocol for inter-processor communications used in distributed, microprocessor-based nuclear safety systems.

The last chapter describes the instantiation of *Isabelle as a theorem proving component for VDM-SL*. The instantiation supports proof in VDM including handling difficult constructs such as patterns and cases expressions in a novel way using reversible transformations. The chapter illustrates the use of the theorem prover on two examples which demonstrate the handling of the three-valued Logic of Partial Functions underlying VDM-SL.

Together these case studies cover a broad range of issues in the use of proof in formal software development in many critical application areas. The book as a whole will be of value primarily to practitioners of formal methods who have experience of writing formal specifications and who need to construct proofs about them. This particularly applies to those seeking conformance with the higher levels of systems development standards, such as U.K. Defence Standard 00-55, the CEC's Information Technology Security Evaluation Criteria, U.S. Department of Defense Standard 5200.28-STD. Secondly, the book should be of use to potential users of formal methods seeking knowledge of existing experience in the application of formal methods and proof. Thirdly, it will be of interest to students of formal methods requiring a more detailed introduction to the practicalities of proof than that provided by the standard tutorial texts and researchers interested in the role of theorem proving in formal development and relevant tool support.

The editor would like to thank the contributors and publisher for their patience and his colleagues, in particular, Brian Ritchie, Stuart Robinson, Kevin Lano and Tom Maibaum, for their support during its preparation.

Contents

Contributors

Sten Agerholm
Institute of Applied Computer Science, (IFAD)
Forskerparken 10,
DK-5230, Odense M, Denmark.
sten@ifad.dk

Juan Bicarregui
Rutherford Appleton Laboratory,
Chilton, Didcot, Oxon OX11 OQX, UK.
j.c.bicarregui@rl.ac.uk

John Fitzgerald
Centre for Software Reliability,
University of Newcastle upon Tyne,
Newcastle NE1 7RU, UK.
John.Fitzgerald@ncl.ac.uk

Jakob Frost
Department of Information Technology,
Technical University of Denmark,
DK-2800 Lyngby, Denmark.

Albert Hoogewijs
Department of Pure Mathematics and Computer Algebra, University of Gent,
Galglaan 2, 9000 Gent, Belgium.

Cliff Jones
Harlequin plc,
Queens Court,
Wilmslow Road,
Cheshire, SK9 7QD, UK.

Peter Lindsay
Software Verification Research Centre,
School of Information Technology,
University of Queensland,
Brisbane, Qld 4072, Australia
pal@it.uq.edu.au

Savitri Maharaj
Department of Computing Science and Mathematics, University of Stirling,
Stirling FK9 4LA, UK.
savi@cs.stir.ac.uk

Brian Matthews
Rutherford Appleton Laboratory,
Chilton, Didcot, Oxon OX11 OQX, UK.
b.m.matthews@rl.ac.uk

Paul Mukherjee
School of Computer Studies,
University of Leeds, Leeds LS2 9JT, UK.
paulm@scs.leeds.ac.uk

Noemie Slaats
Department of Pure Mathematics and Computer Algebra, University of Gent,
Galglaan 2, 9000 Gent, Belgium
ns@cage.rug.ac.be

Bart Van Assche
Department of Pure Mathematics and Computer Algebra, University of Gent,
Galglaan 2, 9000 Gent, Belgium