

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 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	xv
1 A Tracking System	
<i>John Fitzgerald and Cliff Jones</i>	1
1.1 Introduction	1
1.2 Context of the Study	2
1.3 A Formal Model of a Tracking System	3
1.4 Analysing the Model with Proof	10
1.4.1 Levels of Rigour in Proof	11
1.4.2 Validation Conjectures	12
1.4.3 Container Packing	12
1.4.4 Safety of Compaction	17
1.5 Issues Raised by the Study	24
1.5.1 Review Cycle	24
1.5.2 Scope of System	25
1.5.3 Tools	25
1.5.4 Genericity and Proofs	26
1.5.5 Testing as a Way of Detecting Problems	27
1.6 Conclusions	27
1.7 Bibliography	28
2 The Ammunition Control System	
<i>Paul Mukherjee and John Fitzgerald</i>	31
2.1 Introduction	31
2.2 The Specification	32
2.2.1 Explosives Regulations	32
2.2.2 The Model	32
2.2.3 Storing Objects	34
2.3 Satisfiability of <i>ADD-OBJECT</i>	37

2.3.1	Main Satisfiability Proof	39
2.3.2	Formation of the Witness Value	42
2.3.3	Satisfaction of Postcondition	48
2.3.4	Summary	50
2.4	Modifying the Specification	50
2.4.1	Modification to the Specification	51
2.4.2	Proving Equivalence	53
2.5	Discussion	60
2.6	Bibliography	61
2.7	Auxiliary Results	62
3	Specification and Validation of a Network Security Policy Model	
	<i>Peter Lindsay</i>	65
3.1	Introduction	65
3.1.1	Background and Context	65
3.1.2	Software System Requirements	66
3.1.3	Security Threats and Security Objectives	67
3.1.4	Conceptual Model of the Security Policy	67
3.1.5	The Security Enforcing Functions	70
3.1.6	Specification and Validation of the SPM	71
3.2	The Data Model	71
3.2.1	Partitions	71
3.2.2	Users and User Sessions	72
3.2.3	Classifications	72
3.2.4	Messages	72
3.2.5	Seals	73
3.2.6	Sealing	74
3.2.7	Changing Seals	75
3.2.8	Other Integrity Checks	75
3.2.9	Content Checks	76
3.2.10	Accountability Records	77
3.2.11	The Message Pool	77
3.3	The System State	77
3.4	Operations Modelling the SEFs	78
3.4.1	The Authorise Message Operation	79
3.4.2	The Internal Transfer Operation	80
3.4.3	The Export Operation	81

3.4.4	The Import Operation	82
3.5	The Proofs	83
3.5.1	Consistency Proofs	84
3.5.2	Preservation of the Security Properties	85
3.5.3	Completeness Proofs	89
3.6	Conclusions	91
3.7	Bibliography	93
4	The Specification and Proof of an EXPRESS to SQL “Compiler”	
	<i>Juan Bicarregui and Brian Matthews</i>	95
4.1	STEP and EXPRESS	96
4.1.1	The Context	96
4.1.2	The Specifications	97
4.1.3	Related Work	98
4.1.4	Overview	98
4.2	An Outline of EXPRESS	98
4.2.1	Entities	98
4.2.2	Other Type Constructors	100
4.2.3	Subtypes	101
4.3	The Abstract EXPRESS Database	102
4.3.1	The EXPRESS and EXPRESS-I Abstract Syntax	102
4.3.2	The State and Operations	105
4.3.3	Reflections on the Abstract Specification	108
4.4	A Relational Database	109
4.4.1	Signature	109
4.4.2	Datatypes	110
4.4.3	The State and Operations	111
4.4.4	Reflections on the Relational Database Specification	111
4.5	A Concrete EXPRESS Database	112
4.6	A Refinement Proof	113
4.6.1	The Retrieve Function	113
4.6.2	The Refinement Proof Obligations	115
4.6.3	Thoughts on the Refinement Proof	118
4.7	General Experiences and Conclusions	119
4.8	Bibliography	120

5 Shared Memory Synchronization	
<i>Noemie Slaats, Bart Van Assche and Albert Hoogewijs</i>	123
5.1 Introduction	123
5.2 Formal Definitions	124
5.2.1 Program Order and Executions	126
5.2.2 Uniprocessor Correctness	128
5.2.3 Result of a Load	128
5.2.4 Synchronization	130
5.2.5 Memory Model	133
5.3 The VDM Specification of the Definitions	133
5.3.1 Operations	133
5.3.2 Useful Functions	137
5.3.3 The Program Order and Executions	138
5.3.4 Memory Order	143
5.3.5 Uniprocessor Correctness	144
5.3.6 The Result of a Load	144
5.3.7 Synchronization Operations	145
5.3.8 Memory Model	145
5.4 A Theory for Shared Memory Synchronization	145
5.4.1 The Formal Language	145
5.4.2 The Set of Inference Rules	146
5.4.3 A Proof	148
5.5 Discussion	151
5.6 Related Work	152
5.7 Appendix A. A Formal Theory for Relations	153
5.7.1 Signature	153
5.7.2 Axioms	154
5.8 Appendix B. Some Rules Used in the Proof	155
5.8.1 Axioms	155
5.8.2 Proof Obligations	156
5.9 Bibliography	156
6 On the Verification of VDM Specification and Refinement with PVS	
<i>Sten Agerholm, Juan Bicarregui and Savi Maharaj</i>	157
6.1 Introduction	157
6.2 The PVS System	158

6.3	From VDM-SL to the Higher Order Logic of PVS	159
6.3.1	Basic Types, the Product Type and Type Invariants	160
6.3.2	Record Types	161
6.3.3	Sequences, Sets and Maps	162
6.3.4	Union Types	163
6.3.5	Function Definitions	164
6.3.6	Pattern Matching	166
6.3.7	State and Operations	168
6.4	A Specification Example: MSMIE	169
6.4.1	The VDM Specification	170
6.4.2	PVS Translation	173
6.4.3	Typechecking Constraints	175
6.4.4	Some Validation Conditions	177
6.5	Representing Refinement	178
6.5.1	The VDM Specification	178
6.5.2	The PVS Specification	179
6.5.3	The Refinement Relationship	181
6.6	Discussion	183
6.6.1	Using the PVS System	183
6.6.2	Partiality in VDM and PVS	184
6.6.3	Looseness in VDM and PVS	185
6.6.4	Errors in Example Specifications	186
6.7	Conclusion	187
6.8	Bibliography	188
7	Supporting Proof in VDM-SL using Isabelle	
	<i>Sten Agerholm and Jacob Frost</i>	191
7.1	Introduction	191
7.2	Overview of Approach	193
7.2.1	Reading of Figure 7.1	194
7.3	Syntax	195
7.4	Proof System of VDM-LPF	195
7.4.1	Proof Rules	196
7.4.2	Combining Natural Deduction and Sequent Style Proof	196
7.5	Proof Tactics	198
7.5.1	Basic Tactics	198
7.5.2	Proof Search Tactics	199

7.5.3	Gateway Example	201
7.6	Transformations	205
7.6.1	Pattern Matching	205
7.6.2	Cases Expressions	207
7.7	Generating Axioms: An Example	209
7.7.1	Type Definitions	209
7.7.2	Function Definitions	212
7.8	Future Work	216
7.9	Conclusion	217
7.10	Bibliography	219
7.11	VDM-SL Syntax in Isabelle	220

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 0QX, 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 0QX, 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

