FROM FORMAL SPECIFICATION TO FULL PROOF: A STEPWISE METHOD

by

Lavinia Burski

Submitted for the degree of
Doctor of Philosophy

Department of Computer Science
School of Mathematical and Computer Sciences
Heriot-Watt University

March 2016

Abstract

Proving formal specifications in order to find logical errors is often a difficult and labour-intensive task. This thesis introduces a new and stepwise toolkit to assist in the translation of formal specifications into theorem provers, using a number of simple steps based on the MathLang Framework. By following these steps, the translation path between a Z specification and a formal proof in Isabelle could be carried out even by one who is not proficient in theorem proving.

Acknowledgements

I dedicate this thesis to my loving and supportive boyfriend, Jeff.

Contents

List of Tables

List of Figures

Todo list

Acronyms

ASM Abstract state machine.

CGa Core Grammatical aspect.

DRa Document Rhetorical aspect.

GPSa General Proof Skeleton aspect.

Gpsa General Proof Skeleton aspect.

GpsaOL General Proof Skeleton ordered list.

Hol-Z Hol-Z.

IEC International Electrotechnical Commission.

MathLang MathLang framework for mathematics.

PPZed Proof Power Z.

SIL Safety Integrity Levels.

SMT Satisfiability Modulo Theories.

TSa Text and Symbol aspect.

UML Unified Modeling Language.

UTP Unifying theories of programming.

ZCGa Z Core Grammatical aspect.

ZDRa Z Document Rhetorical aspect.

ZMathLang MathLang framework for Z specifications.

Glossary

computerisation The process of putting a document in a computer format.

formal methods Mathermatically rigorous techniques and tools for the specification, design and verification of software and hardware systems.

formalisation The process of extracting the essence of the knowledge contained in a document and providing it in a complete, correct and unambiguous format.

halfbaked proof The automatically filled in skeleton also known as the Half-Baked Proof.

partial correctness A total correctness specification [P] C [Q] is true if and only if, whenever C is executed in a state satisfying P and if the execution of C terminates, then the state in which Cs execution terminates satisfies Q.

semi-formal specification A specification which is partially formal, meaning it has a mix of natural language and formal parts.

total correctness A total correctness specification [P] C [Q] is true if and only if, whenever C is executed in a state satisfying P, then the execution of C terminates, after C terminates Q holds.

Chapter 1

Background

Introduction stating formal methods are a type of mathematics. Explanation of where formal languages came from in mathematics etc

1.1 History of mathematics

Intro....

1.1.1 Right from the begininning

- logic and aristotle
- Frege Grundgesetze + Cantor + Russel discovered paradoxes
- Russel inventing type theory
- Zermelo added axiomisation
- Fraenkel + skolem extended to ZF set theory (which Z is based on)

1.1.2 conclusion

1.1.3 Computerisation of Maths and Proof Systems

1.1.4 Conclusion

1.2 MathLang for mathematics

Intro....

1.2.1 Overview and Goals

1.2.2 Detailed information on CGa

- Reference Zenglars quote

- Weak type theory into CGa

1.2.3 Detailed information on DRa

- relations

- instances

1.2.4 Detailed information on skeletons

1.2.5 information on TSa

1.2.6 A full worked examples in mathlang

1.2.7 Conclusion

1.2.8 History of formal methods and formal languages

I don't know if history of formal methods should be a seperate section or keep it as a subsection here.

- definitions of 'formal language', 'formal method' and 'formal specification'

- the first formal language is thought to be used by Frege in his Begriffsschift (1879), Begriffschift meaning ‘concept of writing’ described as ‘formal language of pure thoguht’
- broad histroy of formal methods
  - 1940’s, Alan Turing annotated the properties of program states to simplify the logical analysus of sequential programs
  - 1960’s Floyd, Hoare and Naur recommended using axiomatic techniques to prove programs meet their specification.
  - 1970’s Dijkstra used formal caluculus to aid development of non-deterinist programs
- Formal methods today
- why use formal methods in industry (design errors like Therac-25 1985, NASAs Checkout Launch and Control System (CLCS) cancelled 9/2002, , added level of rigor)
- types of formal methods (Z, B method, ABS)
- Success of formal methods (B27 Traffic Control System, SHOLIS project, Data Acquisition, Monitoring and Commanding of Space Equipment)
- Weakness of formal methods (Low-level ontologies, Limited Scope,Cost,Poor tool feedback)
- What needs to be done to make formal methods industrial strength?
  - Bridge gap between real world and mathematics
  - Mapping from formal specifications to code (preferably automated)
  - Patterns identified
  - Level of abstraction should be supported
  - Tools needed to hide complexity of formalism

  - Provide visualization of specifications
  - Certain activities not yet formulizable methods
  - No one model has been identified which should be used for software)

ZMathLang covers items 1, 3, 5, 6, 7(semi formal spec)

1.3 Z Syntax and semantics

1.4 Proving systems for Z

1.5 Conclusion

Bibliography

[1] J.-R. Abrial. Event Based Sequential Program Development: Application to Constructing a Pointer Program. In K. Araki, S. Gnesi, and D. Mandrioli, editors, *FME*, volume 2805 of *Lecture Notes in Computer Science*, pages 51–74. Springer, 2003.

[2] J.-R. Abrial. Formal methods in industry: achievements, problems, future. *Software Engineering, International Conference on*, 0:761–768, 2006.

[3] M. Adams. Proof auditing formalised mathematics. *Journal of Formalized Reasoning*, 9(1):3–32, 2016.

[4] A. Álvarez. *Automatic Track Gauge Changeover for Trains in Spain.* Vía Libre monographs. Vía Libre, 2010.

[5] A. W. Appel. Foundational Proof-Carrying Code. In *LICS*, pages 247–256, 2001.

[6] R. Arthan. Proof Power. `http://www.lemma-one.com/ProofPower/index/`, February 2011.

[7] H. P. Barendregt. Lambda Calculi with Types. In *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1991. http://citeseer.ist.psu.edu/barendregt92lambda.htmlElectronic Edition.

[8] B. Beckert. An Example for Specification in Z: Steam Boiler Control. Universitat Koblenz-Landau, Lecture Slides, 2004.

[9] J. C. Blanchette. *Hammering Away, A user's guide to Sledgehammer for Isabelle/HOL.* Institut fur Informatik, Technische Universitat Munchen, May 2015.

[10] E. Borger and R. F. Stark. *Abstract State Machines: A Method for High-Level System Design and Analysis.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[11] N. Bourbaki. *General topology. Chapters 1-4.* Elements of mathematics. Springer-Verlag, Berlin, Heidelberg, Paris, 1989. Trad. de : Topologie gnrale chapitres 1-4.

[12] J. Bowen. Formal Methods Wiki, Z notation. `http://formalmethods.wikia.com/wiki/Z_notation`, July 2014.

[13] A. D. Brucker, H. Hiss, and B. Wolff. HOL-Z 2.0: A Proof Environment for Z-Specifications. *Journal of Universal Computer Science*, 9(2):152–172, feb 2003.

[14] L. Burski. Zmathlang. `http://www.macs.hw.ac.uk/~lb89/zmathlang/`, Jan 2016.

[15] L. Burski. ZMathLang Website. `http://www.macs.hw.ac.uk/~lb89/zmathlang/examples`, June 2016.

[16] R. W. Butler. An introduction to requirements capture using PVS: Specification of a simple autopilot. NASA Technical Memorandum 110255, NASA Langley Research Center, Hampton, VA, May 1996.

[17] R. W. Butler. What is Formal Methods. `http://shemesh.larc.nasa.gov/fm/fm-what.html`, March 2001.

[18] W. Chantatub. *The Integration of Software Specification Verification and Testing Techniques with Software Requirements and Design Processes.* PhD thesis, University of Sheffield, 1995.

[19] Clearsy Systems Engineering. B Methode. `http://www.methode-b.com/en/`, 2013.

[20] J. Coleman, C. Jones, I. Oliver, A. Romanovsky, and E. Troubitsyna. RODIN (rigorous open development environment for complex systems). In *EDCC-5, Budapest, Supplementary Volume*, pages 23–26, Apr. 2005.

[21] I. E. Commission. IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. Technical report, International Electrotechnical Commission, 2010.

[22] E. Currie. *The Essence of Z*. Prentice-Hall Essence of Computing Series. Prentice Hall Europe, 1999.

[23] H. Curry. Functionality in combinatorial logic. In *Proceedings of National Academy of Sciences*, volume 20, pages 584–590, 1934.

[24] C.Weidenbach, D.Dimova, A.Fietzke, R.Kumar, M.Suda, and P. Wischnewski. Isabelle cheat sheet. `http://www.phil.cmu.edu/~avigad/formal/FormalCheatSheet.pdf`.

[25] C.Weidenbach, D.Dimova, A.Fietzke, R.Kumar, M.Suda, and P. Wischnewski. Spass. `http://www.spass-prover.org/publications/spass.pdf`.

[26] N. de Bruijn. The mathematical vernacular, a language for mathematics with typed set. *In Workshop on Programming Logic*, 1987.

[27] L. De Moura and N. Bjørner. Satisfiability Modulo Theories: Introduction and Applications. *Commun. ACM*, 54(9):69–77, Sept. 2011.

[28] D. Fellar, F. Kamareddine, and L. Burski. Using MathLang to Check the Correctness of Specifications in Object-Z. In E. Venturino, H. M. Srivastava, M. Resch, V. Gupta, and V. Singh, editors, *In Modern Mathematical Methods and High Performance Computing in Science and Technology*, Ghaziabad, India, 2016. M3HPCST, Springer Proceedings in Mathematics and Statistics.

[29] D. Feller. Using MathLang to check the correctness of specification in Object-Z. Master Thesis Report, 2015.

[30] Formal Methods Europe, L-H Eriksson. Formal methods europe. `http://www.fmeurope.org/?page_id=2`, May 2016.

[31] S. Fraser and R. Banach. Configurable Proof Obligations in the Frog Toolkit. In *Fifth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2007), 10-14 September 2007, London, England, UK*, pages 361–370. IEEE Computer Society, 2007.

[32] J. Groote, A. Osaiweran, and Wesselius2. Benefits of Applying Formal Methods to Industrial Control Software. Technical report, Eindhoven University of Technology, 2011.

[33] S. L. Hantler and J. C. King. An Introduction to Proving the Correctness of Programs. *ACM Comput. Surv.*, 8(3):331–353, Sept. 1976.

[34] E. C. R. Hehner. Specifications, Programs, and Total Correctness. *Sci. Comput. Program.*, 34(3):191–205, 1999.

[35] A. Ireland. Rigorous Methods for Software Engineering, High Integrity Software Intensive Systems. Heriot Watt Universtiy, MACS, Lecture Slides.

[36] F. Kamareddine and J.B.Wells. A research proposal to UK funding body. Formath, 2000.

[37] F. Kamareddine, R. Lamar, M. Maarek, and J. B. Wells. Restoring Natural Language as a Computerised Mathematics Input Method. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors, *Calculemus/MKM*, volume 4573 of *Lecture Notes in Computer Science*, pages 280–295. Springer, 2007.

[38] F. Kamareddine, M. Maarek, K. Retel, and J. B. Wells. Gradual computerisation/formalisation of mathematical texts into Mizar. In *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*, pages 81–95. Springer-Verlag, 2007.

[39] F. Kamareddine, M. Maarek, and J. B. Wells. Toward an Object-Oriented Structure for Mathematical Text. In M. Kohlhase, editor, *MKM*, volume 3863 of *Lecture Notes in Computer Science*, pages 217–233. Springer, 2005.

[40] F. Kamareddine and R. Nederpelt. A refinement of de Bruijn's formal language of mathematics. *Logic, Language and Information*, 13(3):287–340, 2004.

[41] F. Kamareddine, J. B. Wells, and C. Zengler. Computerising mathematical texts in MathLang. Technical report, Heriot-Watt University, 2008.

[42] Khosrow-Pour and Mehdi, editors. *Encyclopedia of Information Science and Technology*. IGI Global,, 2 edition.

[43] S. King, J. Hammond, R. Chapman, and A. Pryor. Is Proof More Cost-Effective Than Testing? *IEEE Trans. Software Eng.*, 26(8):675–686, 2000.

[44] Kolyang, T. Santen, and B. Wolff. *Theorem Proving in Higher Order Logics: 9th International Conference, TPHOLs'96 Turku, Finland, August 26–30, 1996 Proceedings*, chapter A structure preserving encoding of Z in isabelle/HOL, pages 283–298. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[45] Kolyang, T. Santen, B. Wolff, R. Chaussee, I. Gmbh, and D.-S. Augustin. Towards a Structure Preserving Encoding of Z in HOL, 1986.

[46] A. Krauss. Defining Recursive Functions in Isabelle/HOL , 2008.

[47] R. Lamar. The MathLang Formalisation Path into Isabelle – A Second-Year report, 2003.

[48] R. Lamar. *A Partial Translation Path from MathLang to Isabelle*. PhD thesis, Heriot-Watt University, 2011.

[49] R. Lamar, F. Kamareddine, and J. B. Wells. MathLang Translation to Isabelle Syntax. In J. Carette, L. Dixon, C. S. Coen, and S. M. Watt, editors, *Calculemus/MKM*, volume 5625 of *Lecture Notes in Computer Science*, pages 373–388. Springer, 2009.

[50] P. G. Larsen, N. Battle, M. Ferreira, J. Fitzgerald, K. Lausdahl, and M. Verhoef. The overture initiative integrating tools for vdm. *SIGSOFT Softw. Eng. Notes*, 35(1):1–6, Jan. 2010.

[51] I. Lee, J. Y.-T. Leung, and S. H. Son. *Handbook of Real-Time and Embedded Systems*. Chapman & Hall/CRC, 1 edition, 2007.

[52] K. R. M. Leino. Dafny: An Automatic Program Verifier for Functional Correctness. In E. M. Clarke and A. Voronkov, editors, *LPAR (Dakar)*, Lecture Notes in Computer Science, pages 348–370. Springer, 2010.

[53] M. Lindgren, C. Norstrm, A. Wall, and R. Land. Importance of Software Architecture during Release Planning. In *WICSA*, pages 253–256. IEEE Computer Society, 2008.

[54] I. E. U. Ltd and H. . S. Laboratory. A methodology for the assignment of safety integrity levels (SILs) to safety-related control functions implemented by safety-related electrical, electronic and programmable electronic control systems of machines. Standard, Health and Safety Executive (HSE), Mar. 2004.

[55] M. Maarek. Mathematical documents faithfully computerised:the grammatical and text & symbol aspects of the MathLang framework, First Year Report, 2003.

[56] M. Maarek. *Mathematical documents faithfully computerised: the grammatical and test & symbol aspects of the MathLang Framework*. PhD thesis, Heriot-Watt University, 2007.

[57] M. Mahajan. Proof Carrying Code. *INFOCOMP Journal of Computer Science*, 6(4):01–06, 2007.

[58] M. Mihaylova. ZMathLang User Interface Internship Report. Internship Report, 2015.

[59] M. Mihaylova. ZMathLang User Interface User Manual. Intern User Manual, 2015.

[60] G. C. Necula and P. L. 0001. Safe, Untrusted Agents Using Proof-Carrying Code. In G. Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 61–91. Springer, 1998.

[61] I. C. Office. International Electrotechnical Commission. `http://www.iec.ch/`, July 2016.

[62] S. Owre, S. Rajan, J. Rushby, N. Shankar, and M. Srivas. PVS: combining specification, proof checking, and model checking. In R. Alur and T. A. Henzinger, editors, *Computer-Aided Verification, CAV '96*, number 1102 in Lecture Notes in Computer Science, pages 411–414, New Brunswick, NJ, July/August 1996. Springer-Verlag.

[63] R. L. Page. Engineering Software Correctness. *J. Funct. Program.*, 17(6):675–686, 2007.

[64] B. C. Pierce. *Types and Programming Languages.* MIT Press, Cambridge, MA, USA, 2002.

[65] W. R. Plugge and M. N. Perry. American Airlines' "Sabre" Electronic Reservations System. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '61 (Western), pages 593–602, New York, NY, USA, 1961. ACM.

[66] K. Retel. *Gradual Computerisation and Verification of Mathematics: MathLang's Path into Mizar.* PhD thesis, Heriot-Watt University, 2009.

[67] A. Riazanov and A. Voronkov. The design and implementation of vampire. *Journal of AI Communications*, 15(2/3):91–110, 2002.

[68] G. Rossum. Python Reference Manual. Technical report, Python Software Foundation, Amsterdam, The Netherlands, The Netherlands, 1995.

[69] M. Saaltink and O. Canada. The Z/EVES 2.0 User's Guide, 1999.

[70] S. Schulz. E–a brainiac theorm prover. *Journal of AI Communications*, 15(2/3):111–126, 2002.

[71] J. M. Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[72] M. Spivey. Z Reference Card. `https://spivey.oriel.ox.ac.uk/mike/fuzz/refcard.pdf`. Accessed on November 2014.

[73] M. Spivey. Towards a Formal Semantics for the Z Notation. Technical Report PRG41, OUCL, October 1984.

[74] M. Spivey. The fuzz manual. *Computing Science Consultancy*, 34, 1992.

[75] S. Stepney. A tale of two proofs. In *BCS-FACS third Northern formal methods workshop, Ilkley*, 1998.

[76] I. UK. *Customer Information Control System (CICS) Application Programmer's Reference Manual.* White Plains, New York.

[77] University of Cambridge and Technische Universitat Munchen. Isabelle. `http://www.isabelle.in.tum.de`, May 2015.

[78] Z. Wen, H. Miao, and H. Zeng. Generating Proof Obligation to Verify Object-Z Specification. In *Proceedings of the International Conference on Software Engineering Advances (ICSEA 2006), October 28 - November 2, 2006, Papeete, Tahiti, French Polynesia*, page 38. IEEE Computer Society, 2006.

[79] A. Whitehead and B. Russell. *Principia Mathematica.* Number v. 2 in Principia Mathematica. University Press, 1912.

[80] J. Woodcock and A. Cavalcanti. A tutorial introduction to designs in unifying theories of programming. In *Integrated Formal Methods*, pages 40–66. Springer, 2004.

[81] J. Woodcock and J. Davies. *Using Z: Specification, Refinement, and Proof.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[82] C. Zengler. MathLang- Towards a Better Usability and Building the Path into Coq, First Year Report. Technical report, Heriot-Watt University, November 2008.