ISSN:0975-9646

Formal Specification of UML Use Case Diagram - A CASL based approach

Bhaswati Mondal[#1], Barun Das[*2], Prasenjit Banerjee[@3]

[#1]Department of Computer Science, Vidyasagar University, Midnapore, West Bengal, India
*[*2]Department of Mathematics, Sidho Kanho-Birsha University, Purulia, West Bengal, India*
[@3]Department of Computer Science, Midnapore College, Midnapore, West Bengal, India,

Abstract— **Unified Modelling Language (UML) gives a modelling approach to design a system. Use Case diagram is one of the behavioural approach of UML which describe the behavioural pattern of the system. It has been observed that UML diagrams are not formally specified. Formal specification gives a specific way to design a system as a whole using mathematical notation. This paper proposed formal specification of UML use case diagram using the Common Algebraic Specification Language (CASL) in object oriented paradigm.**

Keywords— **Unified Modelling Language (UML), Use Case diagram, formal specification, non-formal models, Common Algebraic Specification Language (CASL), Object oriented Software.**

I. Introduction

The Unified modelling language (UML) [1], [8], [38] has become a de-facto standard notation for analysis and design models of object oriented software system. It has been observed that graphical representation of model is easily accessible and understandable to the user. The primary gap between the developer and the user, has been easily fulfilled by the graphical description. In UML, Use Case diagram defines the behaviour of a system. It is a visual tool representation that helps the system's end user to understand the behaviour of an element. UML have a well-defined, fully explored semantics which is required in order to ensure that the UML concepts are precisely stated and defined.

In "traditional" engineering like electrical and civil engineering always made their development based on better mathematical technical [2]. So, the validation of these engineering are more perfect and errorless. But the term formal methods that rely on mathematical representation of software, specification analysis and proof, mathematical logics, program verification are not so much familiar to software engineering. If the formal methods can be used in software engineering then it can discover a way to find out an errorless system specification in an unambiguous way. In critical system such as air traffic control information system, railway signalling system, spacecraft systems, have very high validation cost and the costs of system failure are large and increasing. Formal methods can help in this matter by reduce these costs. Formal methods [39] use mathematical notations to precisely express requirements specification. The formal specification removes ambiguity which is inherently present in natural language specification. It also addresses the software reliability and also can effectively improve system reliability, design time and comprehensibility.

The Common Algebraic Specification Language (CASL) [3] is an expressive language for the formal specification of functional requirements and modular design of software. It describes various contexts, subsorts, partial functions, first-order logic, structured and architectural specifications. It also facilitates interoperability of many existing algebraic prototyping and verification code.

The paper has been organized in seven parts. In Section 2, the previous related research on the equivalent domain have been summarized. In Section 3, several element of use case diagram in UML has been discussed using CASL and the case study has also been done to illustrate the model in this section. In Section 4, the future work of the work has been discussed and also concludes.

II. Related Research

Many researches have been proposed towards direction of UML modelling. This section has been organized those proposals with two perspectives. There are many researches [4], [5], [6] have been proposed using UML but these are Non-formal models. Whereas the researches [7, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 40, 41] have been proposed using UML and also formally specified.

A. *Non-formal model using UML*

In this section the published research works that are about non-formal model using UML have been discussed.

In [4], an AUTomotive Open System Architecture in shortly AUTOSAR model has been proposed which is based on architectural component and their interoperability and also supports both client-server and sender-receiver communications. As the main focus of this model is only based on architecture not possible and no design of modelling construct for component based system has also been discussed.

In [5], a component model has been proposed called KobrA (KOmponent Basic Rite Anwendungsentwicklung). It is a based on UML representation. In KobrA the components are not physical components but logical building blocks of the software system. The model provides client-server architecture for communication. This model does not provide any support for EFP and it also does not support any formal specification.

In [6] Palladio component model has been proposed which provides a domain specific modelling language for component based system. This proposal has been expect to increase the performance of the early life-cycle. It defines its own metamodel specified in EMF/Ecore. The components and their role can be connected via assembly connectors to build an assembly. This model also supports client-server architecture for communication. UML has been used to specify the interface in this proposal. However this model does not support any formal specification.

B. Formal model using UML

Under this section the published research work have been discussed about formal model using UML.

In [7], a structured-expandable format of a use case has been proposed which is expressed in Z notation because Z language is a formal specification language. Then it is represented visually using an Entity-relationship diagram. The implemented approach would bridge the gap between a formal language, which is mathematical and UML use case diagram that is visual and used widely to capture requirements. The main goal is when a tool will developed using this approach it will produce a visual representation of a formalized UML use case diagram, from which automated traceability and verification of the design phase. The z-notation is a mathematical notation, specification language and a model-based notation. Z-specification can serve as a single, reliable, reference point for those who investigate the customer's need, test results, type errors in much the same way that a compiler checks code in an executable programming language. It cannot be compiled into a running program. And also another main thing is that Z is a way by which a specification can be decomposed into some small pieces called schemes.

In [22], a result has been shown on formalizing UML behavioural diagram in B notation. It present automatic transformation schemes from UML behavioural diagrams to B specification with the help of UML specification as a tool. B specification has been developed much easily. B is a formal software development where the theoretical aspects of the methods such as the formulation of proof obligations, have done automatically. For these reason B has been adapted for large scale industrial projects. The UML and B is an appropriate combination of object-oriented techniques and formal methods which can give a high quality, errorless and a perfect practical approach of software development. In UML-B integration UML has been controlled by B specification as B supports powerful tools and B specification has become easier with the help of UML specification. This means they are both highly supportive to one another. But B has been still not so easy to learn and use due to its high cost. It can also be said that it is not user friendly.

In [41], the article explains how Aspect-UML models can be specified within the Alloy and how aspect interactions can therefore be verified. That means the work that has been considered is Aspect-oriented (A-O) models written in Aspect-UML. Alloy [36, 35] is a structural modeling language, based on first order logics and designed for the specification of object models through graphical and textual structures. It is based on the ideas of Z [37]. A-O programming provided mechanisms to capture and execute crosscutting concerns in software applications to improve modularity, reuse and maintainability. A-o helps developers to modify the behavior of a base program by introducing aspects which effects at various points on crosscutting throughout a program. But it is difficult to predict the effect of a given aspect on this base program. A-O concepts provide aspects, advices, point cuts, joint points and crosscutting dependencies. It also allows formal annotations such as pre and post conditions. Alloy provides a simple model specification language based on first order logic as well as a model analysis and simulation tool [37].

In [40], a way has been proposed which can create a bridge between informal and formal specification. Most projects have an informal description which can be understood by all people and also there is a need of some formal requirements which can make a project more errorless and perfect. The formal specification can improve the informal specification understanding of the system by exposing gaps and ambiguities in the formal specification. Object constraints Language (OCL) has been used to present the use case form of informal requirements into more formal specifications. But at the development point OCL constraints is not so easy and clear. Because there are so many questions that who is going to add them to the UML diagrams: customers, analysts or designers? Who should understand OCL? That is why OCL is hard to use in industry.

The work proposed in [24, 23], described the translations between Object-Z and UML class diagrams. A metamodel of Object-Z has provided for the benefit of modellers unfamiliar with this formal language. The syntax has been defined by the metamodels, and a model transformation language is dedicated to define the translation. Unfortunately, even this recent work only addresses a subset of the class diagram fragment of UML. The work aims to enable formal verification of UML models, but as yet we have no demonstration nor descriptions of specific techniques.

Model Driven Architecture [25] aims to enable the simultaneous use of many languages, each with syntax defined in MOF, by using model transformations between these languages. The real contribution of [23] is in recognising that formal languages can also participate in this way. Definitive formal semantics could be provided by a Z (or Object-Z) metamodel and UML to Z model transformation. This would enable tool integration, and provide insight into the formalism for the more advanced models. Attempts to directly translate diagrams into formal languages usually ignore the metamodel definition of the language.

The paper [26] advocates an integration of UML and formal methods, in which a UML class diagram is translated into the formal specification language Z. The Z specification is then manually refined, adding details not expressible using class diagrams. The rules and guidelines for semi-automatic translation, they hope, will give insights for developing a more precise semantics for UML.

In the paper [27], it also made integration of a formal language, in this case Object-Z, into the development process. The Object-Z specification manually derived from the class diagram also specifies the class operations. The class is further constrained by a protocol state-machine, which together with the Object-Z schema. They consider several notions of consistency and study which of these are preserved under CSP notions of refinement. However, the intended semantics of the UML fragment are captured by this translation. It is also not clear that the CSP notions of refinement are applicable.

The paper [28] proposes semantics which reconcile the apparently conflicting parts of the UML definition. These semantics concern associations, their ends and the read, create, and destroy link actions. In an appendix to the report, it gives an example model to illustrate the controversy, and expresses his semantics for it in Z. This is intended merely as a precise statement of the proposal explained in the body of the paper. However, this is the most convincing example we have seen of using Z to express dynamic aspects of UML. It is also a good example of why Z will never be widely used by developers: it is not easy to read.

Algebraic specification extended with “generalised labelled transition systems” is used by the paper described in [29] is to formalise parts of UML. It made this by translating UML diagrams into the language CASL-LTL, though it emphasises that the particular language is immaterial. This work explicitly aims for a way of giving useful formal semantics to the whole of UML, and as the title suggests, they take seriously the idea that the different diagrams combine to specify a single system. However they ignore the fact that the official definition already interprets the variety of diagrams into a single abstract syntactic entity, the model.

The Object Constraint Language (OCL) [30] is very much like the languages of traditional symbolic logic, and at least two groups have attempted to make it precise by translating it into well understood systems of logic, intending to enable theorem proving about models.

In the paper [42] use higher order logic (HOL) as implemented in the generic interactive theorem prover Isabelle. The paper [31] uses first order logic. OCL 2.0 has a third truth value “undefined” and allows collections of collections, so first order logic will probably not suffice to formally define it. Neither group make use of the OCL metamodel in their translations. It offers different, equivalent translations optimised for readability or for automated theorem proving respectively. With a foundation as suggested in these works, OCL itself could be the target formal language for a model transformation defining the semantics of UML. This would probably require additions to the current limited temporal operators of OCL though.

The OCL formalisation of the paper [31] is used in the the key project [34]. This is a tool for the deductive verification of Java-Card programs using a specialised dynamic logic [32]. This logic is implemented in a generic theorem prover integrated with the Together modelling tool, and thus provides a practical platform integrating UML modelling and formal methods. Although this work is not aimed at improving the definition of UML, it is instructive. The deductive rules symbolically execute the Java-Card program, and thus give a clear and precise account of the language semantics. The rules could even provide educational interactive animations of the language. Unlike Java-Card, UML is non-deterministic and has no main procedure, but it is conceivable that one could develop such a dynamic logic for UML. The logic would have rules for each of the UML actions. This would define model dynamics, and the meaning of each of the diagrams could be expressed by translation into the dynamic logic language. It would also enable deductive verification of UML models. In its traditional form, dynamic logic is even less readable than Z. But a UML specific logic could use OCL notation for its static parts, whilst the program parts would be written using the yet to be fixed standard UML action language.

The paper [33] uses a formal language derived from dynamic logic to give formal semantics for parts of UML class and state machine diagrams. It defines as a system is a black box, which responds instantly to external stimuli. It is not possible for example to make sense of a sequence diagrams in such a system. This might be a useful interpretation of UML for requirements engineering, as these authors see it, but from our perspective, it is inventing a new language rather than providing a better definition of the existing one.

Since neither Z nor B provides any architectural specification and algebraic prototyping, so a formal specification language is strongly needed which satisfy all these criteria. Common Algebraic Specification Language (CASL) fits for these criteria solving the problem domain.

III. Proposed work

Though UML defines Use Case but it does not provide any strict format or style for defining Use Case. This paper has proposed a structured modelling elements with the corresponding graphical notation to describe the format of Use Case.

A. *Structural modelling elements and corresponding graphical notation*

Use Case Name: It defines the name of the Use Case. Use Case Name contains a character or alpha numeric value to describe the name of the Use Case. It can be graphically represented as a rectangular box .

Id: An Id of the use case defines the unique identification of a Use Case. It contains a numeric or an alpha numeric value to uniquely identify the Use Case. The Id is defined as free type in CASL.

free type Id ::= **sort** int /**sort** Nu
int ::= 0/1/2/……../n
Nu ::= **sort** char ^ **sort** int
char ::= a/b/……./z

Description: It describe whole thing i.e. all works done by the Use Case. It has been described as sentence in CASL. It is a total functions of the sort char. So in CASL it can be described as-

op desc : char → char

Where desc is a total function between the sort characters.

Actor: It defines the player or the performer by whom the use case may be performed. The Actor can be graphically represented as .

Relationship: It defines the different relationships among the different use cases. There are basically two types of relationships-*provided* and *required. Provided relationship* describe that the use case provides few services or a single service to another use case and *required relationship* provides the requirement of the use case from another use case. We define relationship as *total function* and *provided and required relationships* as *partial function*s. A *total function* REL can be defined as follows-

op REL: PR →RD

where PR & RD are *partial functions* defined as

op PR: PR< REL

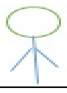
op RD: RD< REL

The graphical representation of the relationship can be define as ──►.

Service: It describes different workings or methods performed by a use case. A service S contains○d as *free type*, relationship as *total function (TF)* and *partial functions (PF)* or operations, description as *sentence*. So, a service can be defined as a *signature* S= {free type, (TF, PF), sentence). A service can be graphically represented as a circle.

The structural modelling elements and their graphical representations are shown in Table 1.

TABLE I: USE CASE STRUCTURAL MODELLING ELEMENTS AND THEIR GRAPHICAL NOTATION

Use Case Structural modelling elements	Graphical Notations
Use Case Name	▭
Actor	
Relationship	──►
service	○

B. CASE Study

In this section, a case study has been done to depict the real life problem like Library Management System. Here the Library Management System has been described partially. Only the Book Management (BM) has been discussed as example. The BM contains few relationships as *book-issue, re-issue, book-return*, *registration, cancel-registration etc.* It also contains few services as *ESI, EBI, and EBN etc.*

***Use Case Name**: Book-Management*

Id: id1

***Description:** The system describes about the relationship of*

the student of the book.

***Actor:** student, librarian*

***Relationship:** book-issue, member-validated, book-validated*

***Service:** {Enter student id / Enter book id / Check whether the book is available or not / Issue the book}.*

The corresponding UML diagram has been shown in figure 1.

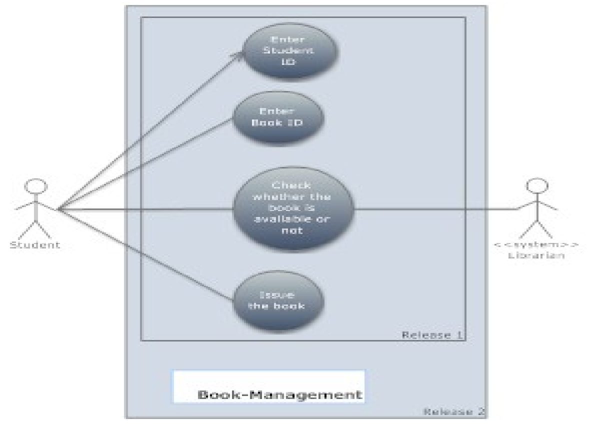


Figure 1. UML Use Case Diagram Book-Management

Each part of the above *Book-Management* has been defined in CASL as follows-

Here service *Enter student id* can be defined in CASL as-

ESI:= {S1, (*Book-issue, member-validated, book-validate*), desc}

Where S1 is the ***free type*** and it defines as-

free type S1::= **sort** Nu

Book-issue is a *total function* and *member-validated & book-validate* are *partial functions.* It can be defined as-

op Book-issue: member-validate → book-validate

op member-validate: member-validate < Book-issue

and **op** book-validate: book-validate < Book-issue.

desc can be defined as **sort** (N1) ::= {Enter member details}

sort (N2)::= {Enter book details}

op *desc:* N1 → N2

IV. FUTURE WORK AND CONCLUSIONS

This paper has proposed the formal specification of the use case diagram of UML. This paper also proposed a set of structural modeling elements and their graphical notations. Using the Common Algebraic Specification Language (CASL), this use case model of UML has been formally specified. A case study has been done to depict the problem domain as library management system. Furthermore, it can be validate using a CASE tool in future.

REFERENCES

[1] Grady Booch, James Rumbaugh, Ivar Jacobson; The Unified Modeling
Language User Guide (1999).

[2] I. Sommerville,, *Formal Specification: book chapter 27, 2009, available at http://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web Chapters/PDF/Ch_27_Formal_spec.pdf.*

[3] Egidio Astesiano, Michel Bidoit ,H- el. ene Kirchner,Bernd Krieg-Br 1 uckner, Peter D. Mossese, Donald Sannella Andrzej Tarlecki., "CASL- the Common Algebraic Specification Language", E.

Astesiano et al. / Theoretical Computer Science 286 (2002) 153 – 196

[4] AUTOSAR Development Partnership, "AUTOSAR – Technical Over- view v2.0.1", 27/06/2006, Available at http://www.autosar.org/download/AUTOSAR_TechnicalOverview.pdf.

[5] C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Paech, J. Wüst and J. Zettel, Component-based Product Line Engineering with UML, Addison-Wesley, 2002.

[6] S.Becker, H.Koziolek,R. Reussner, "Model-Based Performance Prediction with the Palladio Component Model", Proc. of the 6th International Workshop on Software and Performance, Buenos Aires, Argentina, 2007.

[7] S. Sengupta, S.Bhattacharya (2006), "Formalization of UML Use Case Diagram – A Z Notation Based Approach", International Conference on Computing & Informatics (ICOCI) Kuala Lumpur, Malaysia,6 – 8 June 2006.

[8] OMG: Unified Modeling Language Specification, version 2.0. Available at http://www.omg.org/uml

[9] Smith G.O., The Object-Z Specification Language. Advances in Formal Methods. Kluwer Academic Publishers (2000).

[10] Nuno Am´alio, Susan Stepney, and Fiona Polack, Modular UML Semantics: Interpretations in Z Based on Templates and Generics, FACS'03 Workshop on Formal Aspects of Component Software, Pisa, Italy, September 2003

[11] Xuede Zhan, Huaikou Miao, Ling Liu, Formalizing the Semantics of UML Statecharts with Z*, The Fourth International Conference on Computer and Information Technology (CIT'04) 09 14 - 09, 2004

[12] Xiaoping Jia, A Pragmatic Approach to Formalizing Object-Oriented Modeling and Development, COMPSAC, 1997

[13] Jing Sun, Jin Song Dong, Jing Liu, Hai Wang, Object-Z Web Environment and Projections to UML, 10th International World Wide Web Conference (WWW-10), May 2001.

[14] David Roe, Krysia Broda, and Alessandra Russo, Mapping UML Models incorporating OCL Constraints into Object-Z, Technical Reports - Computing - Imperial College London, 2003

[15] Nuno Amalio and Fiona Polack, Comparison of Formalism Approaches of UML Class Constructs in Z and Object-Z, 2003

[16] Margot Bittner, Florian Kamm¨uller, Translating Fusion/UML to Object-Z, Technical University of Berlin, June 2003.

[17] Sophie Dupuy-Chessa and Lydie du Bousquet, Validation of UML models thanks to Z and Lustre, Proceedings of the International Symposium of Formal Methods Europe on Formal Methods for Increasing Software Productivity, 2001

[18] M. Shroff, R.B. France, Towards a formalization of UML class structures in Z, COMPSAC '97 - 21st International Computer Software and Applications Conference 08 11 - 08, 1997 Washington, DC

[19] Soon-Kyeong Kim, David Carrington, Roger Duke, A Metamodel-based transformation between UML and Object-Z, Queensland IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01) 09 05 - 09, 2001 Stresa, Italy

[20] S.-K. Kim, D. Carrington, An integrated framework with UML and Object-Z for developing a precise and understandable specification: the light control case study, Seventh Asia-Pacific Software Engineering Conference (APSEC'00) 12 05 - 12, 2000 Singapore

[21] Soon-Kyeong Kim and David Carrington, A Formal Mapping between UML Models and Object-Z Specifications, Proceedings of the First International Conference of B and Z Users on Formal Specification and Development in Z and B, 2000

[22] Thomas Tilley, Towards an FCA based tool for visualizing formal specifications, ICCS 2003, The 11th International Conference on Conceptual Structures, July 2003, Dresden
Hung LEDANG, Jeanine SOUQUIÈRES, "Formalizing UML Behavioral Diagrams with B", available at www.researchgate.net/...Formalizing_UML_behavioral_diagrams_with_B.

[23] Soon-Kyeong Kim, Damian Burger, and David A. Carrington. An MDA approach towards integrating formal and informal modeling languages. In FM, pages 448–464, 2005.

[24] Soon-Kyeong Kim and David A. Carrington. A formal mapping between UML models and object-Z specifications. In Proceedings of the First International Conference of B and Z Users on Formal Specification and Development in Z and B, pages 2–21, 2000.

[25] Joaquin Miller and Jishnu Mukerji. MDA guide. Technical report, Object Management Group, 2003. http://www.omg.org/mda.

[26] Jean-Michel Bruel and Robert B. France. Transforming UML models to formal specifications. In Proceedings of the OOPSLA'98 Workshop on Formalising UML, 1998.

[27] Holger Rasch and Heike Wehrheim. Checking consistency in uml diagrams: Classes and state machines. In FMOODS, pages 229–243, 2003.

[28] D. Milicev. On the semantics of associations and association ends in UML. Technical report, University of Belgrade, School of Electrical Engineering, February 2006.

[29] Gianna Reggio, Maura Cerioli, and Egidio Astesiano. Towards a rigourous semantics of UML supporting its multiview approach. In H. Hussmann, editor, FASE 2001, volume 2029 of LNCS, pages 171–186. Springer, 2001

[30] Object Management Group. OCL 2.0 specification. Technical report, Object Management Group, 2005. http://www.omg.org/docs/ptc/05-06-06.pdf.

[31] Bernhard Beckert, Uwe Keller, and Peter H. Schmitt. Translating the object constraint language into first-order predicate logic. In Proceedings of VERIFY, Workshop at Federated Logic conferences (FLoC), 2002.

[32] [32]. Bernhard Beckert. A dynamic logic for the formal verification of java card programs. In Java on Smart Cards: Programming and Security, number 2041 in LNCS, pages 6–24. Springer, 2001.

[33] Roel Wieringa and Jan Broerson. Minimal transition system semantics for lightweight class and behaviour diagrams. In Manfred Broy, Derek Coleman, Tom S. E. Maibaum, and Bernhard Rumpe, editors, Proceedings PSMT'98 Workshop on Precise Semantics for Modeling Techniques. Technische Universitaet Muenchen, TUM-I9803, April 1997.

[34] Wolfgang Ahrendt, Thomas Baar, Bernhard Beckert, Richard Bubel, Martin Giese, Reiner H¨ahnle, Wolfram Menzel, Wojciech Mostowski, Andreas Roth, Steffen Schlager, and Peter H. Schmitt. The KeY tool. Software and System Modeling, 4(1):32–54, 2005.

[35] Alloy Homepage. http://alloy.mit.edu.

[36] D. Jackson. Alloy: A leightweight object modeling notation. In MIT Laboratory for Computer Science: Cambridge, MA, 2000.

[37] 17. J. Spivey. The Z Notation. Prentice-Hall,1992.

[38] A. Evans, R. France, K. Lano, B. Rumpe. Devoloping the UML as a Formal Modelling Notation. Available at https://wwwbroy.in.tum.de/publ/papers/EFLR98b.pdf

[39] Siti Halimah Bakri, Hanis Harun, Amera Alzoubi and Rosziati Ibrahim. THE FORMAL SPECIFICATION FOR THE INVENTORY SYSTEM USING Z LANGUAGE, Proceedings of the 4th International Conference on Computing and Informatics, ICOCI 2013 28-30 August, 2013 Sarawak, Malaysia. Universiti Utara Malaysia (http://www.uum.edu.my), Paper No. 064.

[40] Martin Giese and Rogardt Heldal. From Informal to Formal Specification in UML, Chalmers University of Technology, Gothenburg, Sweden. Available at folk.uio.no/martingi/pub/uml04.pdf

[41] Farida Mostefaoui, Julie Vachon, Verification of Aspect-UML models using Alloy, Workshop AOM '07, March 12-13, 2007 Vancouver, British Columbia, Canada Copyright 2007 ACM 1-59593-658-5/07/03... $5.00.

[42] Achim D. Brucker and Burkhart Wolff. A proposal for a formal OCL semantics in Isabelle/HOL. In V.A Carren no, C. Mu noz, and S. Tahar, editors, TPHOLS 2002, volume 2410 of LNCS, pages 99–114. Springer-Verlag, 2002.