Verification Techniques for LOTOS

U. Martin, University of St. Andrews, Scotland, U.K.
M. Thomas, University of Glasgow, Scotland, U.K.

Abstract. We report on the results of a project which applied LOTOS to safety-critical case studies, determined the verification needs of real life applications, and developed tools for ASN.1 to LOTOS translation and equational reasoning.

Verification Techniques for LOTOS [ISO:8807] was a collaborative project, from 1989 to 1993, funded by the UK SERC/IED research program.[1] The project involved four partners: British Telecom PLC, Rutherford Appleton Laboratory, University of St Andrews (the lead partner) and University of Glasgow.

The aim of the project was to investigate the verification requirements of LOTOS specifications, and to determine the applicability of equational reasoning and term rewriting to discharging those requirements. In this paper we summarise some of the major achievements of the project.

The first section contains a summary of our results in five main areas. The second section contains an overview of the case studies developed for the project and the third section contains an overview of the some aspects of the verification and validation requirements which were considered in the project. Software development is discussed in the fourth section and in the final section we conclude with some comments on collabaration within and outwith the project.

1 Summary of Results

1.1 Case studies

Major case studies were undertaken in safety-critical application areas in collaboration with researchers from end user organisations including:

- A medical information bus, in collaboration with Royal Free Hospital School of Medicine.
- A control device for a radiation machine, carried out in collaboration with DEC/SRC Palo Alto.
- A secure login protocol in collaboration with a major UK defence contractor.
- GKS, graphical kernel system, in collaboration with numerous end-users.

These studies extended the use of LOTOS, commonly thought of as appropriate only for specifying protocols, by showing how it could be used in a variety of safety critical applications. They also showed the importance of formal specifications for uncovering errors. Further details are given in Section 2.

[1] IEATP/SE/IED1/4/1477

1.2 Verification requirements

A clear understanding was obtained of the diverse verification requirements that different applications of LOTOS may generate, gained from both theoretical analysis and investigation of case studies. Further details are given in Section 3.

We concluded in particular that it is often the simplest of verification requirements, such as animation, prototyping or correct translation, that are the most important to users, and that any methodology should give some guidance as to what to do if the verification requirements are not satisfied. These conclusions seem to be in line with those of Craigen et al [25].

1.3 Discharge of verification requirements

A clear understanding was obtained of the most appropriate way to use tools in the discharge of these requirements. Further details are given in Section 3.2.

Again our results are in line with [25]: tools should be simple and provide automated support of common decision procedures, together with useful guidance when proofs fail.

In particular the Larch Prover, LP [15], was used in the discharge of requirements in two of the main studies described above, and proved ideal for the rapid development and debugging of proofs.

Our own MERILL system [16] incorporates experimental techniques devised more specifically to discharge verification requirements.

1.4 The ASN.1 to LOTOS translator

The ASN.1 to LOTOS translator is a software tool for translating data type specifications written in the language ASN.1 into LOTOS. The project was carried out by Dr. M. Thomas at the University of Glasgow at the request of the standards group at British Telecom.

This project proved to be a successful application area for formal methods: formalisation of ASN.1 revealed several inconsistencies and omissions from the language design, and functional programming as a prototyping tool enabled quick and effective communication between language designers, implementors, and users. The translator is not only the first formally specified tool for ASN.1, but it is also the first translator for the complete language.

Further details are given in Section 4.1.

1.5 The MERILL prover

MERILL is an equational reasoning theorem prover based on first order term-rewriting. It is highly reconfigurable with an advanced user interface. It supports order sorted logic, which allows the succinct representation of many complex problems.

MERILL incorporates original work in the areas of termination, divergence, order-sorted logic, and completion strategies.

A full description is given is Section 4.2.

2 Case studies

2.1 Medical Information Bus

The Medical Information Bus [3] is a proposed family of standards intended to set forth specifications and guidelines for the definition of a Local Area Network for the interconnection of computers and medical devices. In this case study LOTOS was used to specify one of the protocols from the MIB and the Larch Prover LP to carry out automated verification.

The essential activities which are captured in this case study are the services which an LLC entity must provide to higher layer users, together with how these services are implemented using the services provided by lower layer protocol entities. Thus two specifications are presented and the verification problem is that of demonstrating that these separate viewpoints are consistent. This is done automatically in LP by reasoning with a set of axioms for observational congruence. Since the MIB is not formally specified, one contribution of the work is formalisation of these standards and this inevitably involved resolving ambiguities in the natural language descriptions. The work was carried out jointly by researchers at St Andrews and the Royal Free Hospital School of Medicine.

2.2 Login protocol

A common problem in the verification of any system is to prove a desired relationship holds between a *given* specification and implementation, i.e. when the implementation has *not* been derived from the specification directly. This case study [8] considers just such a problem, where the specification, implementation and verification requirements where given at the start. The problem is interpreted here as a login protocol (since the real end-user application is confidential). The specification, implementation and verification requirement have first to be written in LOTOS as the original problem statement has some informal features.

An initial attempt at proving equivalence between the specification and the implementation failed because key information is missing from the specification. The information is added in a modular way (as a constraint), so that the original specification is unaltered, and eventually the verification requirement is proven to be satisfied. Some of the proofs are automated using the RRL rewriting system tool. The work was a collaboration between University of Glasgow and a major defence contractor.

2.3 Radiotherapy machine

The high-level behaviour of a linear accelerator machine, used for radiotherapy, is specified in LOTOS. Some important safety properties concerning radiation overdoses are formalised (by grammars) and then property testing, trace analysis, temporal logic, and other reasoning techniques are used to develop a correct design. Much of the property testing is carried out using the LOTOS symbolic

transformation tool LOLA, although some of the proofs have still to be finished by hand. This case study [19] is motivated by the overdosing accidents involving the Therac-25 linear accelerator (a computer controlled radiotherapy machine involved in several accidental fatal overdoses in the U.S.A. and Canada in the late 1980s). Various high level aspects of that machine's behaviour such as the interaction with the user are explored. This case study represents a new application area for LOTOS: design for safety-critical systems.

2.4 Incorrectness of Software from the Therac-25

This work [20] involves the attempted verification of a particular piece of assembler pseudocode concerning data entry and machine initialisation for the above-mentioned Therac-25 linear accelerator. The code is taken from a published technical account of some features of the (faulty) machine which has also been part of a legal submission in the court cases following the accidents involving this machine. The code is formally specified (reverse engineered) as an equational specification and some correctness properties are defined for the pseudocode. The equational reasoning theorem prover LP is then used to try and verify the code against its specification. One important proof fails, but in such a way as to indicate how the behaviour of the pseudocode causes the failure. This leads to a modification of the code, and ultimately, correctness proofs for the modified code. This work is the first known formalisation of any aspect of the Therac-25 code or design and it is also a good demonstration of how formal methods can be used reveal errors. The work was carried out while the author was a visiting researcher at DEC/SRC Palo Alto.

2.5 GKS - graphical kernel system

The complexity of interactive graphics functionality requires compact and precise description techniques for their design and to study their properties. The graphical kernel system, GKS, was first published as an ISO/IEC standard in 1985 and is now going undergoing revision, as GKS-R. Since GKS-R combines both process and data views of computer graphics, this is a good application area for LOTOS. Accordingly, in this case study LOTOS is used to specify the input and output graphics functionality of a considerable subset of GKS-R [6, 7].

2.6 Pass the parcel

This small case study [22] considers specifying a children's game: pass the parcel, in LOTOS. A published LOTOS specification of the game (Bustard et al [2]) is studied and found to be insufficiently detailed to specify the full behaviour implied by the informal description. A further, amended specification is proposed and tested using LOLA.

2.7 Conclusions

Our main conclusions are:

1. LOTOS is commonly thought of as a language for specifying protocols. We showed it could be used in new application domains, such as graphics standards and high integrity systems such as the radiotherapy machine case study.
2. We were using real end-user applications and not applications developed for this project.
3. We were able to show the importance of developing and understanding a specification for:
   - identifying and resolving ambiguities in English specifications, e.g. the login case study,
   - identifying errors or omissions in published formal specifications, e.g. the pass the parcel game case study,
   - identifying errors in a specification with respect to an implementation, e.g. the login case study,
   - giving the first formal description of a known faulty high integrity system and using it to find some of the faults, e.g. the radiotherapy machine case study.
4. We found, particularly in the review case studies, a strong role for formal methods in uncovering errors, rather than for verifying correctness per se. Further, the way in which the errors were revealed informed us as to how to make corrections, e.g. in the radiotherapy machine and the Therac-25 software case studies. This role seemed to be independent of the actual formal methods used.
5. We uncovered the influence of the specification and implementation style on the verification requirements, e.g. in the login case study.

3 Verification and validation requirements for LOTOS

A main objective was to investigate the verification and validation requirements for LOTOS, and what tool support was necessary to discharge them.

We characterise validation as the convincing demonstration of conjectures which may involve application of tests, simulation, and exploration of a model. Verification is the formal, rigorous proof of properties of the specification, or specifications under inspection. Essentially, both involve analysis, where validation may be regarded as non-exhaustive and verification just a special case of validation.

Our attention has been focused on 6 main aspects.

Translation Correct translation between ASN.1 and LOTOS was identified both by British Telecom and within the MIB project. The translator is described in 4.1.

Animation and prototyping One of the chief ways of identifying errors is through testing and experimentation using prototyping and animation tools. Whilst non-exhaustive testing cannot prove a general case, testing can increase confidence that a good model has been constructed, that a conjecture might be a theorem, and also to reveal errors. A testing method particular to LOTOS is *property* testing. Property testing is a form of state reachability analysis in which one quantifies over a class of states, expressed as a LOTOS process. So, whilst it is a form of testing, it is a much more abstract and powerful technique than testing by simulation.

Experimentation was implicit in all of the case studies, but it was most extensively carried out and the benefits felt in the login and radiotherapy machine case studies (particularly property testing in the latter), and in the course of the work on the ASN.1 to LOTOS translator.

Equivalences and satisfaction Equivalence and satisfaction requirements have traditionally been seen as the main focus of verification for LOTOS, i.e. proving that a particular relation holds between two specifications such as an observational congruence, or testing congruence. This approach is particularly useful when one specification is to be regarded as the implementation or abstraction of another.

A good deal of effort has to be put into understanding the difficult theory underpinning the various equivalences and into determining how they relate to the actual verification requirements for a particular system. These issues are considerably more complex for LOTOS because of the interconnections between the data type and process parts. The particular difficulties of relating specifications with different data/process boundaries had not previously been given much attention, although an identified problem in the LOTOS community,

Criteria for choosing between the equivalences with respect to the specific needs of LOTOS are given in [9] and applied in the login case study.

Alternative formalisms Alternative formulations of verification requirements were identified in the login case study, particularly the need to express and prove *temporal* properties. Moreover, whilst some temporal properties can be expressed within the LOTOS framework, it may be more natural to express a specification constraint or a verification requirement in a temporal logic. Preliminary investigations into this area are given in the radiotherapy machine case study and in [9].

Specification style LOTOS specifications can be written in a variety of styles, e.g. monolithic, resource oriented and constraint oriented. The chosen style (possibly a mixture for different components of a system) may influence a final implementation, or at least the ease with which it is constructed. We have shown that it also affects the verification process, particularly with respect to modularity and extensibility, as demonstrated in the login case study.

Errors Verification and validation are as much concerned with successful demonstration of conjecture and theorem as with lack of success. It is easy to fail to prove a desired correctness property and this poses difficult questions: how long do we persevere to gain an acceptable demonstration, what is the measure of acceptability, and when should we give up? These key questions were raised (and, to some extent, answered) in the review case studies. It is clear from these studies that experience of the verification process and extensive testing are a prerequisite to answering these questions.

3.1 Conclusions

1. The simplest kinds of verification/validation requirements: correct translation, prototyping and animation, seemed to be the easiest to achieve and the most important to users.
2. Equivalence/satisfaction and temporal verification requirements are necessary for a more complete approach but they are much harder to understand and to automate.
3. It is very important to understand why and how the verification process fails; is it because the property does not hold, or because there is insufficient theory to demonstrate it?

3.2 Tool support

We used three kinds of tool: the ASN.1 translator, equational reasoning provers and process algebra manipulation tools.

Equational reasoning tools such as LP [15], RRL [27] and MERILL (Section 4.2) performed well on *animation and prototyping* and *non-correctness and errors*.

We experimented with them for *equivalence and satisfaction proofs* as well. Proving equivalence is after all the traditional domain of equational reasoning, but modelling the equivalences we needed proved problematic and required elaborate induction proofs even in a simplified model. Therefore we used more specialised tools such as PAM [28], the Process Algebra Manipulator, and LOLA [29], a LOTOS symbolic simulation tool.

4 Software development

Two major pieces of software development were carried out under the project: the development of the ASN.1/LOTOS translator and the further development of the MERILL equational reasoning prover.

4.1 The ASN.1/LOTOS translator

The ASN.1 to LOTOS translator [21] is a software tool for translating data type specifications written in the language ASN.1 into LOTOS. Since ASN.1 is

a well established language (with an ISO/CCITT standard) in the field of data communications, there is a need to translate the significant body of protocol specifications already written using ASN.1, and, further, to allow engineers to continue developing the data types for protocols using ASN.1. These types can then be translated and integrated with LOTOS process descriptions.

The translator was formally specified by giving a rigorous semantics defining the relationship between the two languages. The semantics evolved through four distinct versions since 1989, each of which was implemented by a translator prototype.

The semantics was the first *formal* semantics for ASN.1, and since it was the first, the prototypes were an essential aid to communicating the full consequences of the chosen semantics. The prototypes were written in a functional programming language (Miranda) with good algebraic types and so the relationship between the formal semantics and the implementation (i.e. the prototype translator) was a very clear and simple one. Thus, changes to the semantics were quickly and correctly implemented. Furthermore, feedback from engineers and ASN.1 language designers using a prototype could be easily reflected in the semantics.

This type of project proved to be a successful application area for formal methods: formalisation of the language revealed several inconsistencies and omissions from the language design, and functional programming as a prototyping tool enabled quick and effective communication between language designers, implementors, and users.

The translator is not only the first formally specified tool for ASN.1, but also the first translator for the complete language.

4.2 The MERILL prover

As mentioned above the MERILL system [16, 17] was chosen as our experimental inference engine. The starting point for this new system was the ERIL system designed and implemented in Prolog by Jeremy Dick between 1982 and 1989 [4]. It has been developed under this project by Brian Matthews and the first stage of development was the redesign of this system and re-implementation in Standard ML. This new system provided not only an order of magnitude increase in efficiency but a much cleaner design upon which to build extra functionality essential to the application area. This system now represents one of the most advanced systems of its type currently available.

It is highly reconfigurable with an advanced user interface. This gives it a considerable advantage when experimenting with new reasoning or search strategies: the user can easily experiment rather than be tied to the limited built in possibilities of older systems.

It supports order sorted logic, which allows the succinct representation of many complex problems. However this is not just syntactic sugar; it also supports reasoning techniques for order sorted rewriting and completion.

Further techniques developed under the project included new techniques for proving termination [12, 14], for handling divergent sets of rewrite rules [23, 24],

and for dynamic typing [17].

5 End-User Collaboration

An important part of the project was collaboration with end-users through the case studies mentioned above, in particular Royal Free Hospital School of Medicine British Telecom, Institute of Radiation Oncology, Western Infirmary, Glasgow, Department of Aerospace Engineering, University of Glasgow, DEC/SRC Palo Alto/Massachusetts Institute of Technology. Many individuals also contributed: Juan Bicarregui, Adam Cichon, Dave Cohen, David Duce, Carron Kirkwood, Mike Lai, Brian Matthews, Kathy Norrie, Chris Reade, Brian Ritchie, Elizabeth Scott, Mike Spivey and Phil Watson.

To obtain the MERILL system contact Brian Matthews: `bmm@inf.rl.ac.uk`.

The full project report may be obtained from Prof U Martin at the Department of Computer Science, University of St Andrews.

References

1. J.C. Bicarregui, Algorithm Refinement with Read and Write Frames, Proceeding of Formal Methods Europe '93 Symposium, Odense, April 1993, LNCS 670, Springer-Verlag.
2. D.W. Bustard, M.T.Norris, R.A. Orr, and A.C. Winstanley, An Exercise in Formalizing the Description of a Concurrent System, Software practice and experience, 22 (1992).
3. P. Curran and K.J. Norrie, An approach to verifying concurrent systems - a medical information bus (MIB) case study, *in* Proceedings of the 5th annual IEEE symposium on computer-based medical systems, Durham, North Carolina, 74 – 83, 1992.
4. A. J. J. Dick and J. R. Kalmus, ERIL Users manual, Rutherford Appleton Laboratory 1988, RAL-88-055
5. A. J. J. Dick and P. Watson, Order-Sorted Term Rewriting, Computer Journal, 34 (1991) 16–19
6. D.A. Duce and F. Paterno, A Formal Specification of a Graphics System in the Framework of the Computer Graphics Reference Model, Computer Graphics Forum, 12 (1993) 123-130,
7. D.A. Duce and L. Damnjanovic, Formal Specification in the Revision of GKS: An Illustrative Example, Computer Graphics Forum, 11 (1992) 17-30
8. C. Kirkwood, Automating (Specification = Implementation) using Equational Reasoning and LOTOS. *in* TAPSOFT'93: Theory and Practice of Software Development, Springer Lecture Notes in Computer Science 668, (1993) 544–558
9. C. Kirkwood, Verification of LOTOS Specifications Using Term Rewriting Techniques, Ph.D. thesis, University of Glasgow, 1994.
10. U. Martin, A geometrical approach to multiset orderings, Theoretical computer Science 67 (1989) 37-54
11. U Martin, Linear interpretations by counting patterns, *in* Springer Lecture Notes in Computer Science 690, 5th International Conference on Rewriting Techniques and Applications, Montreal, 1993

12. U. H. Martin and T. Nipkow, Ordered Rewriting and Confluence *in* Proceedings of the 10th International conference on Computer Aided Deduction, Lecture Notes in Computer Science 607, 1990
13. U. H. Martin and M. K. F. Lai, Some experiments with a completion theorem prover, Journal of Symbolic Computation (1992) 13, 81-100
14. Ursula Martin and Elizabeth Scott, The order types of termination orderings on terms, strings and multisets, *in* Proceedings of the Eighth IEEE Conference on Logic in Computer Science, Montreal, 1993
15. U.H. Martin and J. Wing (editors), Proceedings of the First International Workshop on Larch, Workshops in Computer Science Series, Springer Verlag, 1993.
16. B.M. Matthews, MERILL: An Equational Reasoning System in Standard ML: A User Guide RAL technical report, RAL-93-026, April 1993.
17. B.M. Matthews, MERILL: An Equational Reasoning System in Standard ML, *in* Springer Lecture Notes in Computer Science 690, 5th International Conference on Rewriting Techniques and Applications, Montreal, June 1993
18. E.A.Scott, An automated proof of the correctness of compiling specification, *in* M. Nivat, T. Rus, G. Scollo editors, Proceedings of AMAST: Algebraic Methodology and Software Technology, Workshops in Computing, Springer Verlag, 1993.
19. M. Thomas, The Story of Therac-25 in LOTOS, High Integrity Systems Journal, 1, (1994) 3-15
20. M. Thomas, A Proof of Incorrectness using the LP Theorem Prover: The Editing Problem in the Therac-25. High Integrity Systems Journal, 1 (1994) 35-48
21. M. Thomas, A Translator Tool for ASN.1 into LOTOS. *in* M. Diaz, Ro. Groz, editors, Formal Description Techniques, V. pp. 37 - 52, Elsevier Science Publishers B.V. (North-Holland), 1993.
22. M. Thomas, A Note on An Exercise in the Formalization of a Concurrent System, manuscript, 1993.
23. M. Thomas and P. Watson, Solving Divergence in Knuth-Bendix Completion by Enriching Signatures. Theoretical Computer Science, 112, (1993) 145-185
24. M. Thomas and P. Watson, Solving Divergence in Knuth-Bendix Completion by Enriching Signatures. Extended abstract. *in* M. Nivat, T. Rus, G. Scollo editors, Proceedings of AMAST: Algebraic Methodology and Software Technology, Workshops in Computing, Springer Verlag, 1993.
25. D. Craigen, S. Gerhart and T. Ralston, An International Survey of Industrial Formal Methods. Report NISTGCR 93/626 from US Department of Commerce, National Institute for Standards and Technology, March 1993.
26. ISO [ISO:8807] Information Processing Systems: Open Systems Interconnection: LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, 1988.
27. D. Kapur and H. Zhang, RRL : Rewrite Rule Laboratory User's Manual, 1987.
28. H. Lin, PAM: A Process Algebra Manipulator, University of Sussex, Computer Science Technical Report, 2/91.
29. J. Quemada, S. Pavón, A. Fernandez, Transforming LOTOS specifications with LOLA - The Parameterised Expansion,*in* Formal Description Techniques I, K. Turner (ed.), North-Holland, 1988.

This article was processed using the LaTeX macro package with LLNCS style