Evaluating a Formal Methods Technique via Student Assessed Exercises

Alastair F. Donaldson* and Alice Miller

Department of Computing Science
University of Glasgow
Glasgow, Scotland.
{ally,alice}@dcs.gla.ac.uk

Abstract. We present a case study in evaluating a formal methods technique, using student assessed exercise solutions as sample input to SymmExtractor, a symmetry detection tool for the SPIN model checker. We discuss the ethical procedure which must be followed when using student programs for research, and present the results of our evaluation.

1 Introduction

The mathematical nature of formal methods means that designers of a technique or tool may give little thought to its widespread usability, concentrating chiefly on theoretical soundness and elegance with respect to a limited application domain. Although the rigour of a formal approach depends on these theoretical aspects, usability is clearly important if techniques and tools are to be adopted by industry, educators, or other researchers.

Formal methods students are suitable participants for evaluation of formal methods techniques — their interest is balanced between the practical aspects of a particular tool and the theory which underlies its construction, and they will reject a technique which has limited application even if it is theoretically elegant. Further, such evaluation can be mutually beneficial: students can learn about practical formal methods by using certain techniques and tools, and simultaneously aid research into the usability of prototype extensions.

The contribution of this paper is a case study evaluating an automatic symmetry detection tool, SymmExtractor. The evaluation is based on a set of example solutions to an assessed exercise from the *Modelling Reactive Systems* final year course at the University of Glasgow. We discuss the ethical issues involved in using student programs for research, present the design of our evaluation, and propose some changes to SymmExtractor based on the evaluation results. The benefit of this evaluation was one way: students allowed us to use their solutions in our research. We conclude by proposing a symbiotic approach in which students' education in formal methods can also benefit from participation in evaluation.

* Supported by the Carnegie Trust for the Universities of Scotland.

2 Symmetry in Model Checking

Model checking [5] is a popular automated formal reasoning technique whereby temporal logic properties of a concurrent system can be checked via an abstract, finite state model of the system. A widely used model checker suitable for protocol verification is SPIN, which allows reasoning over specifications written in Promela [10]. The application of model checking is limited due to the state space explosion problem – as the number of components in a system increases, the state space of its associated model grows combinatorially, becoming too large to reason over. Symmetry reduction techniques aim to combat this problem by exploiting replication in the topology of the system. Such replication may induce automorphisms, or *symmetries* of the underlying state space. These automorphisms form a group, any subgroup of which can be used to partition the state space into equivalence classes. Certain temporal properties can be checked over a *quotient* state space, consisting of one state per equivalence class. This can potentially result in large savings in memory and verification time [3, 7].

For details of symmetry reduction in model checking, see a recent survey [12]. We now describe two tools for analysing symmetry in Promela models. The first, SymmExtractor, infers symmetries of the state space underlying a Promela specification via static analysis of the specification. The second, SPIN-to-GRAPE, allows generators for the group of all symmetries of a small state space to be computed, and was used extensively in the design and evaluation of SymmExtractor.

2.1 The SymmExtractor and SPIN-to-GRAPETools

For symmetry reduction to be a useful technique it is vital to be able to detect symmetries of a state space without actually building the state space. There are three main approaches to symmetry detection: restricting the input language so that the state space associated with a specification is *guaranteed* to be symmetric [15]; extending the language with keywords which can be used to *specify* symmetry [11]; and inferring symmetries of the state space by analysing the communication structure of a specification [3, 6]. The third approach is less restrictive than the first, and avoids the manual effort of the second, thus potentially allows symmetry reduction to be a "push button" technique.

SymmExtractor [6] performs automatic symmetry detection for Promela by extracting the *static channel diagram* (SCD) of a Promela specification. The SCD is a graphical representation of potential communication between components of the specification. Generators for the group of symmetries of the SCD, $Aut(SCD)$, is computed, and the computational algebra system GAP [9] is used to determine generators for a subgroup of $Aut(SCD)$ which induces a symmetry group of the underlying state space, and thus can be used for symmetry-reduced model checking. SymmExtractor is incorporated in TopSPIN [7], which uses the detected symmetries to for state space reduction when model checking with SPIN.

SymmExtractor requires that a Promela specification obeys certain restrictions. These include: the use of an `init` process in which all processes are in-

stantiated simultaneously; restrictions on the use of channel variables; and a constrained set of allowable operations on variables which take as their values process identifiers. These restrictions aim to make automatic symmetry detection tractable and straightforward to implement, without unduly modifying the way that Promela programs are specified. The evaluation presented in this paper aims to identify mismatches between these restrictions and natural modelling styles used by students in a set of sample specifications.

For small models, it can be illuminating to construct the state space as a directed graph and explicitly compute its group of automorphisms. The Spin-to-Grape tool [8] outputs the state space underlying a Promela specification as a directed graph suitable for input to Grape [16], a graph-theoretic add on to Gap. Grape can then be used to compute the automorphism group of the state space and the corresponding quotient state space.

Spin-to-Grape can be used to explicitly analyse the symmetry group of a state space with up to several thousand states. The tool was used extensively for testing purposes during the development of SymmExtractor and TopSpin [7] (a symmetry reduction package for Spin which incorporates SymmExtractor). In Section 4.2 we discuss the role of Spin-to-Grape in the evaluation of SymmExtractor.

3 Modelling Task – a Telephone Exchange

Modelling Reactive Systems (MRS) is a final year 20-lecture formal methods course at the University of Glasgow. The primary focus of the course is on the theory and practice of model checking, and students use Spin in practical sessions. The main prerequisite for MRS is a discrete mathematics course for computing science, which covers the basics of set theory, predicate logic, relational algebra and methods of proof. In addition, students are required to have passed first year mathematics courses on calculus and algebra, as well as multiple computing science courses on programming, data structures and algorithms. Almost 20% of the assessment for MRS is via a practical exercise which involves specifying a reactive system using Promela, then reasoning about the specification with Spin.

The MRS practical exercise for 2004/2005 involved producing three versions of a specification for a two user telephone system. Intuitively, a Promela specification of a two-user telephone exchange should exhibit one non-trivial symmetry which switches the local states of the users (and their associated channels) throughout all global states. Thus solutions to this modelling task provide a good set of Promela examples with which to evaluate the restrictions imposed by SymmExtractor, discussed in Section 2.1. Further, the associated state spaces are small enough for Spin-to-Grape to compute all state space symmetries present in a given specification, which can be compared with those detected by SymmExtractor.

4 Evaluation

4.1 Ethical Approval

To ensure that our user study is ethical, we have followed the *Glasgow Ethics Code* check-list [13]. This is a 12-point check-list distilled from the ethical standard of the British Psychological Society [2], and focuses on the issues which are most relevant to computing science projects. Compliance with most of the points on the check-list was straightforward. The following points required some care:

- **All participants explicitly stated that they agreed to take part** Students who allowed us to use their solutions in the study were provided with an information sheet detailing the aims of the study, and asked to sign a consent form (provided as Appendix A and adapted from a standard example [14]). The intended usage of students' solutions is detailed in Section 4.2.
- **The researcher conducting the experiment is not in a position of authority or influence over any of the participants** As the solutions formed part of the course assessment, it was important that the the consent of students was not sought until after solutions had been assessed and returned. This assured students that their decision to take part in the study could have no effect on their score for the exercise, and encouraged them to answer the assessed questions in exactly the same way as they would have otherwise.

A further ethical concern is that the assessed exercise should be designed to meet the intended learning outcomes of the course and not to meet research aims (unless these overlap). In addition, since assessment has been shown to narrow students' focus [1], care must be taken to ensure that an assessment biased towards the research interests of the course director does not restrict breadth of learning. In our case the exercise had been set to meet the course aims before we designed our evaluation.

The study was approved by the ethics committee of the Faculty of Information and Mathematical Sciences at the University of Glasgow (ref. *FIMS00203*). We obtained signed consent forms from 17 students from a class of 35.

4.2 Methods

For each specification in the sample set we gathered the following data by a combination of automatic and manual analysis:

1. Size of the unreduced state space (computed using SPIN)
2. State space symmetries computed by SPIN-to-GRAPE, and size of the resulting quotient state space
3. Symmetry breaking features of the specification, and modifications required to restore symmetry (documented by experimenter)
4. Violations of restrictions imposed by SymmExtractor (as reported by the tool) and modifications required to satisfy restrctions (documented by experimenter)

5. Symmetries detected by SymmExtractor
6. Size of the quotient state space computed by TopSpin.

Symmetry breaking features are aspects of the specification which destroy the intuitive symmetry discussed in Section 3. When Spin-to-Grape showed absence of this expected symmetry in a given specification, the experimenter manually examined the specification to identify symmetry breaking features. We classify the modifications of 4 above as *minor* if they could be avoided by a straightforward extension of SymmExtractor, *medium* if they would be unnecessary if SymmExtractor could capture symmetry between global variables, or *major* if they could only be avoided by significant development of the theory on which SymmExtractor is based. The quotient state space is computed using TopSpin to ensure that it matches the quotient structure constructed with respect to the symmetries computed by Spin-to-Grape.

4.3 Results

We refer to the individual components of a three part solution as specifications. Of the 51 specifications analysed, just over half did not exhibit the expected symmetry due to symmetry breaking features. In most cases this was because `run` statements were not surrounded by an `atomic` block; for other examples the telephone users were initialised asymmetrically (e.g. *handset* variables for users 1 and 2 were set to *up* and *down* respectively, destroying symmetry between users). In all cases it was possible to restore symmetry by trivial modifications, with a negligible effect on the global state space. With these modifications, SymmExtractor was able to detect symmetry immediately from 23 of the resulting specifications. A further 13 required modifications which we classified as *minor* – these included replacing locally instantiated channels with globally instantiated channels, and removing channel instantiation statements from record declarations. Another 7 specifications required *medium* modifications (as described above). The final 8 specifications required *major* modifications. These modifications have identified a problem with the usability of the tool, which involves the way that arrays indexed by process identifiers are accessed.

We have extended the SymmExtractor documentation with a short set of modelling guidelines based on the problems encountered when applying the tool to this set of examples. It is clear that the tool would be more useful if it could handle symmetry between global variables, and an approach to this extension is sketched in [6]. The main challenge which the evaluation results have presented is to find techniques to automatically determine the relationship between numeric identifiers passed as parameters to processes by the user (and used to access arrays), and the runtime id values which Spin assigns to processes.

5 Conclusions and Future Work

We have presented a case study in formal methods evaluation, using solutions to a student assignment as input to the SymmExtractor symmetry detection tool.

The evaluation has identified some necessary improvements to the tool, as well as some modelling styles which destroy potential symmetries of a model. We have also discussed the ethical procedure which we followed before using student programs.

We are currently evaluating SymmExtractor further using another set of student programs, which model a railway signalling system. In future we hope to carry out this kind of evaluation *during* a formal methods course, so that students can learn by critically analysing new add-ons (like our symmetry reduction package) to existing tools. The students were unaware, when designing their solutions to this assessed exercise, that symmetry detection and reduction techniques would later be applied to their programs. However, symmetry reduction is part of the MRS course. Therefore it would be useful to see the effect on their programming style had they been asked to build specifications suitable for symmetry reduction. As development of SymmExtractor and TopSPIN continues this is very much a future research direction.

References

1. J. Bowden, G. Masters and P. Ramsden. Influence of assessment demands on first year students' approaches to learning. *Research and Development in Higher Education: A Forgotten Species?*, pages 397–407. Higher Education Research and Development Society of Australia, 1987.
2. British Psychological Society code of conduct. http://www.bps.org.uk/
3. E.M. Clarke, E.A. Emerson, S. Jha, and A.P. Sistla. Symmetry reductions in model checking. In *CAV'98*, LNCS 1427, pages 147–158. Springer, 1998.
4. E.M. Clarke, R. Enders, T. Filkhorn, and S. Jha. Exploiting symmetry in temporal logic model checking. *Formal Methods in System Design*, 9(1–2):77–104, 1996.
5. E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.
6. A.F. Donaldson and A. Miller. Automatic symmetry detection for model checking using computational group theory. In *FM'05*, LNCS 3582, pages 418–496. Springer, 2005.
7. A.F. Donaldson and A. Miller. A computational group theoretic symmetry reduction package for the SPIN model checker. In *AMAST'06*, LNCS 4019, pages 374–380. Springer, 2006.
8. A.F. Donaldson, A. Miller and M. Calder. SPIN-to-GRAPE: a tool for analysing symmetry in Promela models. *Electronic Notes in Theoretical Computer Science*, 139(1):3–23, 2005.
9. The Gap Group. *GAP–Groups Algorithms and Programming, Version 4.2*. Aachen, St. Andrews, 1999. http://www-gap.dcs.st-and.ac.uk/~gap.
10. G. J. Holzmann. *The SPIN model checker: primer and reference manual*. Addison Wesley, 2003.
11. C.N. Ip and D.L. Dill. Better verification through symmetry. *Formal Methods in System Design*, 9(1/2): 41–75, 1996.
12. A. Miller, A. Donaldson and M. Calder. Symmetry in temporal logic model checking. *Computing Surveys*, 2006. To appear.
13. H.C. Purchase. Student compliance with ethical guidelines: The Glasgow Ethics Code. In *Proc. Higher Education Academy 6th Annual Conference on Information and Computer Sciences*, pages 82–85, 2005.

14. H.C. Purchase. Example participant consent form. http://www.dcs.gla.ac.uk/∼/ethics/
15. A.P. Sistla, V. Gyuris, and E.A. Emerson. SMC: a symmetry-based model checker for verification of safety and liveness properties. *ACM Transactions on Software Engineering and Methodology*, 9(2):113–166, 2000.
16. L.H. Soicher. Computing with graphs and groups. In *Topics in Algebraic Graph Theory*, pages 250–266. Cambridge University Press, 2004.

Appendix A

Participant Consent Form: Symmetry in Promela Models

The aim of this experiment is to investigate structural symmetry arising in typical Promela models of distributed systems.

The experiment will involve allowing the experimenter to analyse your assessed exercise submission for last year's Modelling Reactive Systems 4 course. The analysis is concerned with the structure of the state space underlying your solutions, *not* with the semantic correctness of the solutions.

All results will be held in strict confidence, ensuring the privacy of all participants. No personal participant information will be stored within the data. Data will be stored online in a password protected computer account.

A feedback email message will be sent to all participants, after the data has been analysed.

Please note that it is the Promela language, not you, that is being evaluated. You may withdraw from the experiment at any time without prejudice, and any data already recorded will be discarded.

If you have any further questions regarding this experiment, please contact:

Alastair Donaldson
Computing Science Department
Lilybank Gardens
ally@dcs.gla.ac.uk

I have read the information sheet, and agree to voluntarily take part in this experiment:

Name: _____ Email: _____

Signature: _____ Date: _____

This study adheres to the BPS ethical guidelines, and has been approved by the FIMS ethics committee of The University of Glasgow (ref: FIMS00203). Whilst you are free to discuss your participation in this study with the researcher (contactable on 330 4236 ext. 0049), if you would like to speak to someone not involved in the study you may contact the chairs of the FIMS Ethics Committee: {s.garrod,s.schweinberger}@psy.gla.ac.uk.