Expanding the proof rule base and tactics of AtelierB automated prover - Research Proposal

Agata Borkowska, UID: 1690550, *MSc in Computer Science, University of Warwick*

***Abstract*—AtelierB is a tool for formal software development through refinement, using the B-method. It incorporates an automated prover, which has been recognized as the most thorough prover for B set theory, and has been used as a basis for many others. Nevertheless it has multiple shortcomings. Various approaches have been suggested and taken to improve its performance, including extensions to the proof rule base, created by the users. In this work we aim to create such an extension, ensuring that all added rules are sound and well-reasoned. We also aim to identify any limitations of this approach. The secondary goal is to improve the robustness of the software without straying from pure B method, and taking into account the ease of use. As a metric of our success, we use the benchmarks proposed by Conchon and Iguernala [1].**

***Index Terms*—B method, formal verification**

I. Introduction

THE aim of formal specification and verification is to ensure the correctness of software. While overall less popular than quality assurance through testing, it is most commonly used in safety-critical areas, such as air traffic control, railway routing, or medical cyber-physical systems, or where testing is too costly or puts the users at risk. It allows for certifying that a given piece of software works as intended, and is error-free.

In some industries, such as railway, it is demanded by the regulations to provide a certificate of correctness for any piece of software. While not required, it is strongly suggested to use formal verification methods, such as B or Z, and indeed they make satisfying the regulations easier. [2]

A. The B-method

Various methods and approaches have been developed. In this project, we shall focus on the classical B-method, allowing for formal specification through refinement of abstract machines. In the earliest stages of the development process, an abstract machine is created, that satisfies the requirements. It is a construct based on 1st order logic and set theory. At this point, there is no reasoning about the order of operations and timeline - no loops, no sequential actions. Additionally the machine may, and often will have some nondeterministic elements.

The abstract machine is then refined step by step, until the nondeterminism is eliminated. The final stage of the refinement process is the implementation, which is a much more concrete description of the system, ready to be translated into code. Thus it has to be entirely deterministic, and use simple data types, such as numerical values or arrays, but not sets. While limiting, it brings us very close to the actual coding, and in fact some tools are capable of code generation from the implementation machine.

The B-method allows us to do much more than turning an initial specification into an auto-generated (and because of that often low quality) code. It gives us means of documenting the whole software development process, and certifying that the final product indeed meets the requirements.

B. AtelierB tool

The tool to support all stages of software development with B-method is AtelierB, created by ClearSy. In particular it includes a powerful automated prover. We will inspect it and the proof process very closely, as the work in this project is expected to rely on it entirely.

After creating the specification, we can generate proof obligations (POs). They are statements about the consistency of the types of variables, well-defineness and other things that are implied by our definitions or methods. In a correct piece of software, all of the POs should be demonstrated to hold. If there is a PO that cannot be proved to hold by any means, it indicates that there exists an error in the specification.

An easy example to demonstrate it is a bag, from which we pick an item one by one. The operation to pick an item guarantees some return value, so a proof obligation generated by the tool may require us to check that the bag is never empty when this operation is called.

It is expected that about 75% of the POs will be demonstrated automatically, and only 5% of them will require more than 10 s (i.e. proof force 1 or higher) [3].

The remaining POs will need user input to be discharged. This is done through the automated prover, with the user chosing which rule or tactic to apply, simplifying the hypothesis and in general nudging the prover and pointing it in the right direction.

C. Limitations of the prover

It is important to understand that the prover can only generate proof obligations and demonstrate that they are satisfied, but is incapable of proving that they are not - this problem is undecidable. In other words, each proof obligation can only be proved or unproved, and the latter can happen for one of two reasons. It is quite likely that a PO is true, but the program cannot prove it automatically because of its complexity. However, the unproved obligations also include the ones that are false and point to errors in the specification.

Finally, it has to be stressed that adding user-created rules is not advised, and should be avoided, however that may not always be possible. The tools available are not perfect and additionally may not be tailored to the particular scenario. All added rules must be proven to be correct, before one can rely on them. Although the prover has some capability for demonstrating a rule by reducing and rewriting it into a simpler form, new rules often have to be validated by hand. There is nothing stopping users from adding incorrect rules, and especially ones that do not deal with all possible test cases are often not shown to be false easily.

II. Related Works

A. Applications of B-method

Although it may seem purely theoretical at first, the B-method has industrial applications, especially in safety-critical areas. It is3 commonly used in European Railway industry, from traffic management systems to developing platform door controllers [5].

An example of using classical B-method in a large scale industrial project is given by F. Badeau and A. Amelot [7]. They have aided Siemens Transportation Systems to develop an shuttle rail service at Paris-Charles de Gaulle Airport. The fully automatic metro line began its operation in April 2007, and has been functioning since then, with one more metro line added in June 2007.

This example is the most illustrative one, as the authors provide a clear breakdown of what work went into proving both the abstract and the concrete models. They have generated over 43 thousand proof obligations, 97% of which were proven automatically. While it is a quite good result, at this scale it leaves over 700 POs to be proven by hand, and according to the authors, their average rate was proving manually around 15 obligations per day. It demonstrates just how time consuming it is to work with a tool that is not as automatic as one claims to be.

More importantly, the authors had the need for 290 new proof rules. They have discovered that the Predicate Prover in AtelierB was very useful in validating them. 84% have been done by the tool, however the remaining ones have consumed many man-hours to validate. It is a pity that more details on this are not provided.

In the formal model of railway stations proposed by K. Reichl *et al.* [4], an evolution of B-method called Event-B is used, as it allows for modelling systems as a whole, including hardware. Nevertheless the underpinning theory and relying on first order logic and set theory is common to both B and Event-B. Event-B also requires Rodin tool, rather than AtelierB, however in their work, the authors have used AtelierB prover in addition to Rodin's one.

K. Reichl *et al.* have achieved a similar proportion of proved POs to Badeau and Amelot - 97.6% of their POs were discharged by the prover. This number seems to be quite consistent. E. Bernard *et al.* in their B model of GSM 11-11 smart cards had 377 proof obligations generated, and all but one of them were discharged automatically (the last one requiring the interactive prover) [8]

B. Methods for verifying proof rules

The Atelier B Maintenance edition includes a tool for validating mathematical rules. Unfortunately SMT and deep embedding goes here

C. Plugins, extensions and other software

A very common approach to improving the provers, be it for the classical B-method or Event-B, is to create a plugin or an extension to AtelierB and Rodin respectively. While this is not the aim of the project, and we make an attempt to improve the tool by means available through them, some of the methods are portable and give us an idea of how to approach certain problems. Additionally we look at the problems the creators of the extensions came across, as a warning and inspiration.

A reason for creating extensions and modificatons to the software was discussed by M. Leuschel *et al.* in their paper on case studies of large scale models of metro lines in various cities [9]. They found that AtelierB was not optimised very well and tended to run out of memory, when working on real-life models.

- BWare

- Dedukit
- Beval
- Coq
- Isabelle

D. Shortcomings of Atelier B

As mentioned before, M. Leuschel *et al.* were very critical about Atelier B, and it is worth looking closely at the issues they have raised. Their main problem with the tool and the focus of their paper was lack of optimisation, however they listed others as well. They noted that in the case of undischarged POs, it was "difficult to find out why the proof has failed".

An important warning to be taken out of their paper is concerned with the DEFINITIONS section of an abstract machine. Their behaviour is almost identical to that of macros, and may be prone to errors if not inspected properly. The example given in the aforementioned paper is a definition `sum(x,y) == x+y` which AtelierB will use as is. Thus an expression `sum(1,1)*2` will result in `1+1*2`, which is clearly not the intended outcome. [NOTE: WILL TRY TO REPLICATE ON LAB MACHINES OR PC. LAPTOP's ATELIERB STOPPED WORKING]

The issues mentioned above are quite general and in line with our own experience. We will not go into the details of bugs raised in the paper, as the authors have been using Atelier B version 3.6.2, while the one we use 4.2.1, released in December 2014. According to the release notes, more than 150 bugs have been fixed since the previous version. However, no list of known bugs and issues has been found for the newest version. [10]

Other problems are mentioned by XYZ...

While we will not aim to fix these issues, it is necessary for us to be aware of them.

III. Project Aims

In this project we shall focus on the AtelierB automated prover. The key aim of the project is to push it to its limits, without creating ay plugins or extensions. We are interested to see how much can be achieved with the prover on its own.

We shall also look at different ways of writing and phrasing the specification to make it more prover-friendly. Some of the questions we would like to answer are: are there any ways of expressing commonly found points that will make them faster to prove automatically? How much impact does the complexity of expressions have on the prover's ability to discharge POs? If any of the items found above are non-intuitive, can we create tactics that will bridge the gap between the prover's ability and the user's way of thinking?

A secondary aim of the project is to analyse the user-friendly aspects of the AtelierB tool or lack thereof. The difficulty to understand why a proof has failed was brought up by M. Leuschel *et al.* in their broad case study [9], among others.

It is expected that over the course of this project, new questions and ideas will arise. Some, but not all may be pursued, while others will be identified as areas for further research.

A. Choice of Scenarios

To recognize the most commonly problematic proof obligations, we will collect a few scenarios created by various research groups or companies. This will ensure that the issues with the prover will not be user-specific.

The most important criteria in choosing third party models will be created using only B-method (and not extensions to it, such as Event-B), and that they will be well-documented, especially in terms of tools used for verification.

As it has been identified in the literature review, the industry that most commonly uses the B-method in practice is Railway, and there are models available online [cite]. However, it is well within the scope of this project to assess if there are problems common to different scenarios.

Thus, the railway industry will be the primary focus, however we will compare the improvements made across a range of scenarios, not limited to this area.

B. Metrics

C. Expected learning outcomes

D. Appropriateness of Research Methods

IV. Project Management

A. Methodology

B. Timeline

Key dates, as listed by the CS907 Dissertation Project website, are:

- **19th January:** Registration of dissertation topics
- **16th February:** Submission of project proposals
- **24-28th April:** Project presentations
- **6th July:** Submission of interim reports
- **14th September:** Submission of dissertation

It is also important to take into account dates of terms, which are 9th January to 18thMarch for the Spring Term, and 24th April to 1st July for the Summer term, with the university examinations commencing on or after the

15th May, and being spread over a period of about two weeks. Therefore, it is to be expected that little progress will be made during Spring Term and especially in May, with the bulk of the work being done over holiday and after the examination period.

The Gantt chart in Fig. X shows an estimate of the project's timeline.

C. Progress

D. Constraints and Risks

1) Copyrights for AtelierB software: Atelier B is a closed source project, however it is distributed free of charge, for any purpose including teaching and research. ClearSy does not forbid users from making modifications to their software, however any modified software will be excluded from their maintenance service policy.

The maintenance edition of the software includes a tool for proving mathematical rules. It could potentially be helpful to have access to it in the project, however as of now we do not have the means to gain access to it.

2) Bugs in AtelierB and lack of responses from ClearSy: It should be highlighted that we have not found a list of bugs and known issues with any version of Atelier B. The company blog includes a page on it, however it has not been updated since August 2012, and the content (which could be of some use, even if it were outdated) does not load.[1] ClearSy has been contacted for us to gain access to the updated information.

Thus our main source of informations about issues with the software is academic publications and unfortunately word of mouth. This is not reliable at all. It is also worth pointing out, that despite multiple attempts to contact ClearSy about bugs and other problems found in Atelier B since October last year, we had no response so far.

If any bugs are encountered during the work on this project, they will be reported to ClearSy as well as highlighted in the project.

3) Requirement for knowledge outside the subject area: The B-method relies heavily on first order logic and Zermelo-Fraenkel set theory, and especially the latter is beyond the scope of the course (MSc in Computer Science). Fortunately, this area has been covered in depth during my previous degree (BA in Mathematics).

4) Risk of data loss or machine failure: A GitHub repository has been set up to contain a remote back up of the work done so far, thus also safeguarding against theft or loss of data storage devices. It has the additional benefits of allowing work from multiple machines, and convenient tracking of changes. The address of the repository is: https://github.com/agata-borkowska/dissertation .

This is a reasonable precaution we have deemed it to be sufficient, provided the changes are committed whenever significant progress has been made.

5) Time estimates: This project is intended to be flexible, and following an agile approach. It is fully expected that over the course of this project, new questions will occur, and we may or may not choose to pursue them. Thus, it should be noted that the presented timeline is very rough. There are however a couple of milestones, such as the end of performing a review of existing methods and scenarios, and the beginning of benchmarking, which should not be postponed. To this end, time to work on the project will be scheduled and adhered to, and progress will be reported to the Supervisor.

Additionally, regular meetings with the Supervisor will aid with keeping it on track, Her advice will also be helpful in assessing if the pace of work is sufficient.

Therefore, mistakes in the estimates of time taken to complete tasks are not a severe issue.

V. Concluding remarks

References

[1] S. Conchon and M. Iguernlala, "Increasing Proofs Automation Rate of Atelier-B Thanks to Alt-Ergo" in *Proc. 1st Int. Conf. Reliability, Safety and Security of Railway Systems* (RSSRail 2016), Springer, 2016, pp. 243-253

[2] *Railway applications - Communication, signalling and processing systems*, EN 50128, 2011

[3] *Interactive Prover User Manual*, v. 3.7, ClearSy Sys. Eng., Aix-en-Provence, France, p. 27

[4] K. Reichl et al., "Using Formal Methods for Verification and Validation in Railway" in *Proc. 10th Int. Conf. Tests and Proofs*, Springer, 2016, pp. 3-13

[5] T. Lecomte et al., "Formal Methods in Safety-Critical Railway Systems" in *Proc. Brazilian Symp. Formal Methods* (SMBF 2007), 2007, pp. 26-30

[6] M. Jastram, "Rodin User's Handbook v.2.8", Düsseldorf, 2012

[7] F. Badeau and A. Amelot, "Using B as a High Level Programming Language in an Industrial Project: Roissy VAL" in *Proc. 4th Int. Conf. Z and B Users*, Springer, 2005, pp. 334-353

[8] E. Bernard et al., "Generation of test sequences from formal specifications: GSM 11-11 standard case study" in *Software: Practice and Experience*, vol. 34, Wiley, 2004, pp. 915-948

[9] M. Leuschel et al., "Automated property verification for large scale B models with ProB" in *Formal Aspects of Computing*, vol. 23, Springer, 2011, pp. 683-709

[10] *Atelier B version 4.2 release notes*, ClearSy Sys. Eng., Aix-en-Provence, France, 2014

[1]http://tools.clearsy.com/tools/bug-status/ Note the first comment about the size of the table.