Debugging Support in Atelier B

Léa Riant[0000−0001−8779−527X]

ClearSy System Engineering, Aix-en-Provence, France
lea.riant@clearsy.com

Abstract. Insightful error reports save precious time in the design of systems. When using the formal B method, design errors correspond to invalid proof obligations. The legacy automatic provers in Atelier B are not capable to identify if a failure to prove is due to a logical error. In contrast, SMT solvers are capable to prove a first-order logic formula but also to disprove it and to produce a counter-example. Those counter-examples can give precious indications to the user on design errors. SMT solvers have been integrated in the most recent version of Atelier B, but only to use their proving capabilities. We present here counter_example_reader, a tool to interpret a counter-example produced by an SMT solver into a B counter-example.

Keywords: Atelier B · Debugging · B method · SMT

1 Introduction

The development of critical systems requires a level of safety that can not be obtained by submitting the system to a series of tests that simply cannot be exhaustive. Thus, the use of formal method for critical safety has been democratized.

The B method [1] was defined in the late 80s. It was use industrially for the first time in the development of the automatic pilot of a metro line in Paris. The formal IDE Atelier B was developed in parallel to implement the B method. Atelier B and the B method have been used in many critical systems, mainly in the railway industry.

Atelier B historically has two internal provers. A plugin now allow the use of the ProB model checker [8] in Atelier B. Its prover can manage bigger data structures [7] and return counter-examples. The BWare project [6] develops a framework to use the Why3 [4] plateform on Atelier B's projects. Why3 is a plateform that allow to call different automatic theorem provers. SMT solvers are regularly used in logic-based verification task. They have already been implemented in Rodin [5], another formal IDE. Since its last release, Atelier B 4.7 include the use of SMT solvers.

We present here the continuation of the integration of automated provers in Atelier B by exploiting the possibility to return counter-examples, described in Section 3. It adds a counter-example translator, named counter_example_reader and described in Section 4, and modify the translation of B component into

SMT input. The setup necessary and a brief overview of use will be presented in Section 5. We will mention the limitations of the tools in Section 6.

2 Presenting the B method

A project using the B method starts with mathematical models and ends with computer implementations through refinements. Atelier B generate lemmas (called here proof obligations) to assure the coherence of a model. Then, for each refinement, proof obligations are automatically produced to ensure that the new one is consistent with the previous level. A proof obligation is an implication $H \Rightarrow G$, where H is a set of hypotheses and G is the goal. There can be a quite considerable number of consistency and refinement proof obligations. Real projects can easily have around 160.000 proof obligations. If a great part can be proved by the Atelier B's historical provers, the rest needs to be discharged through an interactive proof's interface, which represent a considerable time. Those interactive proofs are a great part of the development cost.

3 SMT solver

3.1 Reliability of results

Unlike Atelier B's internal provers, external provers are not necessarily certified for critical use. But critical projects must prove that they respect security requirements. Due to that, those solvers are only used as auxiliary tools to provide information but with a certified solver or redundancy, their result could be relied on.

Therefore, the result of an external prover cannot be used to discharge a proof obligation, but it provides useful information as results of an SMT prover have an high probability to be true. In the preliminary stages of a project, a lot of proof obligation cannot be discharge by Atelier B. SMT solvers tend to prove more proof obligations than Atelier B's provers [9]. So SMT provers can be used to sort provable proof obligations from false ones arising from incorrect specifications. Time is not wasted on trying to prove impossible proof obligations. And in the event that an SMT solver provides a counter-example, it can point at problematic cases and speed up the correction of the specification.

3.2 Satisfiability and validity

An SMT solver does not prove the validity of a formula but its satisfiability. It consider a formula as a boolean instance where each atomic predicate is a boolean variable. If the instance possesses a solution, the solver check if the truth values of the predicates are coherent with its theories. If it is coherent, the formula is satisfiable, else the solver search for another solution. In some case, particularly when quantified expressions are involved, the solver cannot prove

that the theories are respected or not, and the solution is considered uncertain. An SMT solver result will be either "sat", "unsat" or "unknown".

Subsequently, we can not use an SMT solver to prove or disprove the original proof obligation. To prove $H \Rightarrow G$, we need to check the satisfiability of its negation, $H \wedge \neg G$.

If the solver cannot build a valuation of the variables of $H \wedge \neg G$ that respect the formula and its theory, the formula is unsatisfiable and $H \Rightarrow G$ is valid.

Otherwise, if the solver can build a valuation within the constraints, $H \wedge \neg G$ is satisfiable, which disprove $H \Rightarrow G$ and the valuation can be return as a counter-example.

If the solver is uncertain of its valuation, the formula will be deemed "Unknown". But the valuation can still be retrieved and used as a possible counter-example.

3.3 Translating to SMT-LIB

Most SMT solvers use the standard language SMT-LIB [2]. To translate a B component into the SMT-LIB format, we adapted a tool already developed by Clearsy, ppTransSmt. The major change was not in the actual translation of a proof obligation but in the additional information that may be needed by the solver, such as an axiomatization of B operators. But such definitions often contain quantified expressions that are not only useless for the proof of the current goal, but that can cause the solver to be inconclusive regarding the satisfiability and to produce the result "Unknown".

4 Integration of external provers

To use of SMT solver and their counter-example functionality, four processes are required. We need a sound (and efficient) translator from B to SMT-LIB, the solver itself, a reader associated to the first translator and a translator from SMT-LIB to B models.

Driver As we can use multiple solvers, translators or readers, we chose an approach similar to Why3 by declaring mechanisms that can contain drivers which describe how Atelier B should interact with each solver and which processes should fulfill each role.

Writer Its role is to translate $H \wedge \neg G$ into SMT-LIB format. We already evoked ppTransSmt, the writer currently used by most drivers in Atelier B to translate proof obligation into SMT-LIB format. There also exist a simplified writer that expresses set operators as uninterpreted symbols but is less accurate as its translation is not equivalent to $H \wedge \neg G$ but a consequence of it.

Prover An SMT solver need to process SMT-LIB and a driver specifying its parameters to allow Atelier B to launch it.

Status reader The reader is the more straightforward process of the four. It changes the status of the result by its interpretation, according to the accuracy of the writer. Typically, for the reader associated with ppTransSmt :

$$sat \Rightarrow Disproved$$
$$unsat \Rightarrow Proved$$
$$unknown \Rightarrow Unknown$$

Counter-example reader The translator counter_example_reader can work with any SMT solver. It browse the counter-example return by the solver line by line and store information on variables. The information are retrieved with regular expressions. Then, the name of the constant is deobfuscate, its value and type are translated to B models by analysing their SMT-LIB description operator by operator. For example, in the Figure 1 :

```
1 Disproved
2 (model
3 (define-fun g_switch_2 () Bool true)
4 (define-fun g_var_a_0 () Int 0)
5 (define-fun g_var_b_3 () Int (- 1))
6 (define-fun g_var_c_4 () Int 1)
7 (define-fun g_var_d_1 () Int 1)
8 (define-fun g_var_e_5 () Int (- 1))
9 )
```

Fig. 1. Counter-example return by the solver cvc4

The third line describe a constant function named g_switch_2 will be stored in a Variable object as :
Variable(”switch”, ”BOOL”, ”TRUE”).

For sets and functions, the name and type are translated but the value is not. ppTransSmt is based on membership functions and the value returned by SMT provers is untranslatable as is.

5 Uses

To use SMT-solver in Atelier B, driver adapted to the solver and the objectives is needed. The use of external prover must be enable, the driver must be selected in the available mechanisms list and the path to the solver must be added to the resources of the project in its settings. The matching button can be add in the visual interface via the option ”Add an external prover”.
You will see that the status of each component will have two more categories : ”Unreliably proved” and ”Disproved”, as in Figure 2.

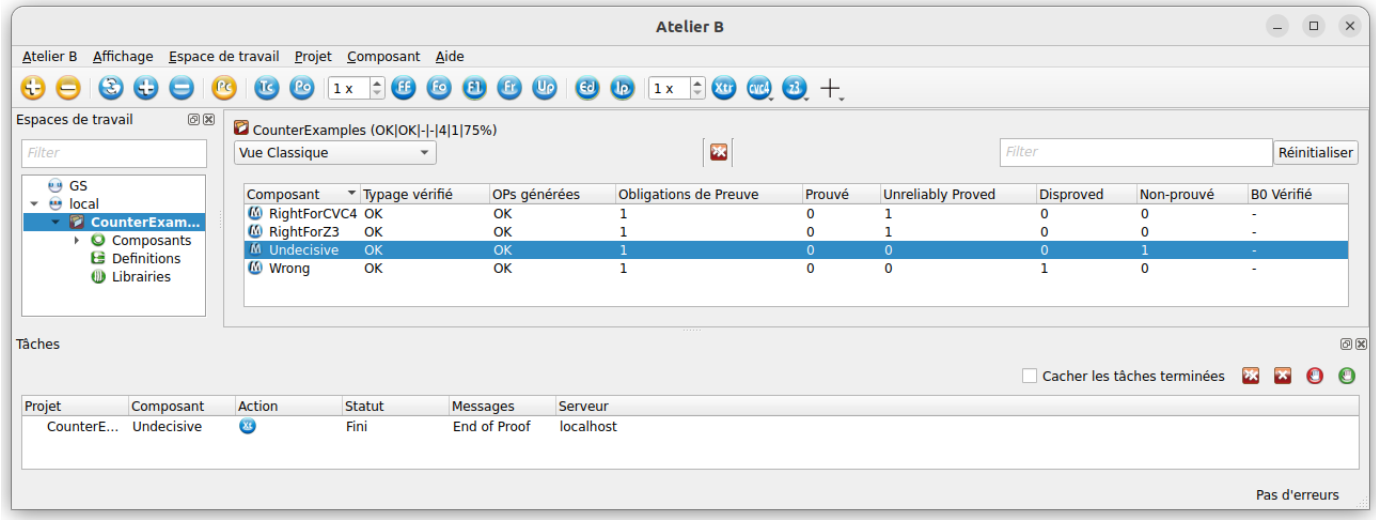

Fig. 2. Atelier B's project management window

5.1 Bbatch

Once the project is open with :

open_project <name>

An external solver can be called in the bbatch terminal :

extprove <component> <mechanism> [option = 0 (all), 1 (fast only)]

And a counter-example can be retrieved :

extcounter_example <component> <po> <mechanism> <driver>

5.2 GUI

When a component is selected, the external provers button will be available. When a mechanism is choosed, Atelier B will try to prove unproved every proof obligation of the component with each driver.
Once a solver has tried to prove a proof obligation and failed (either Disproved or Unknown status), an option is available in the edition window, shown in Figure 3. With a right click on the name of a proof obligation, a list of all drivers having disproved or failed to prove the proof obligation will be unroll. They can produce counter-example and uncertain counter-example respectively, which will be printed in the "Counter-example" window.

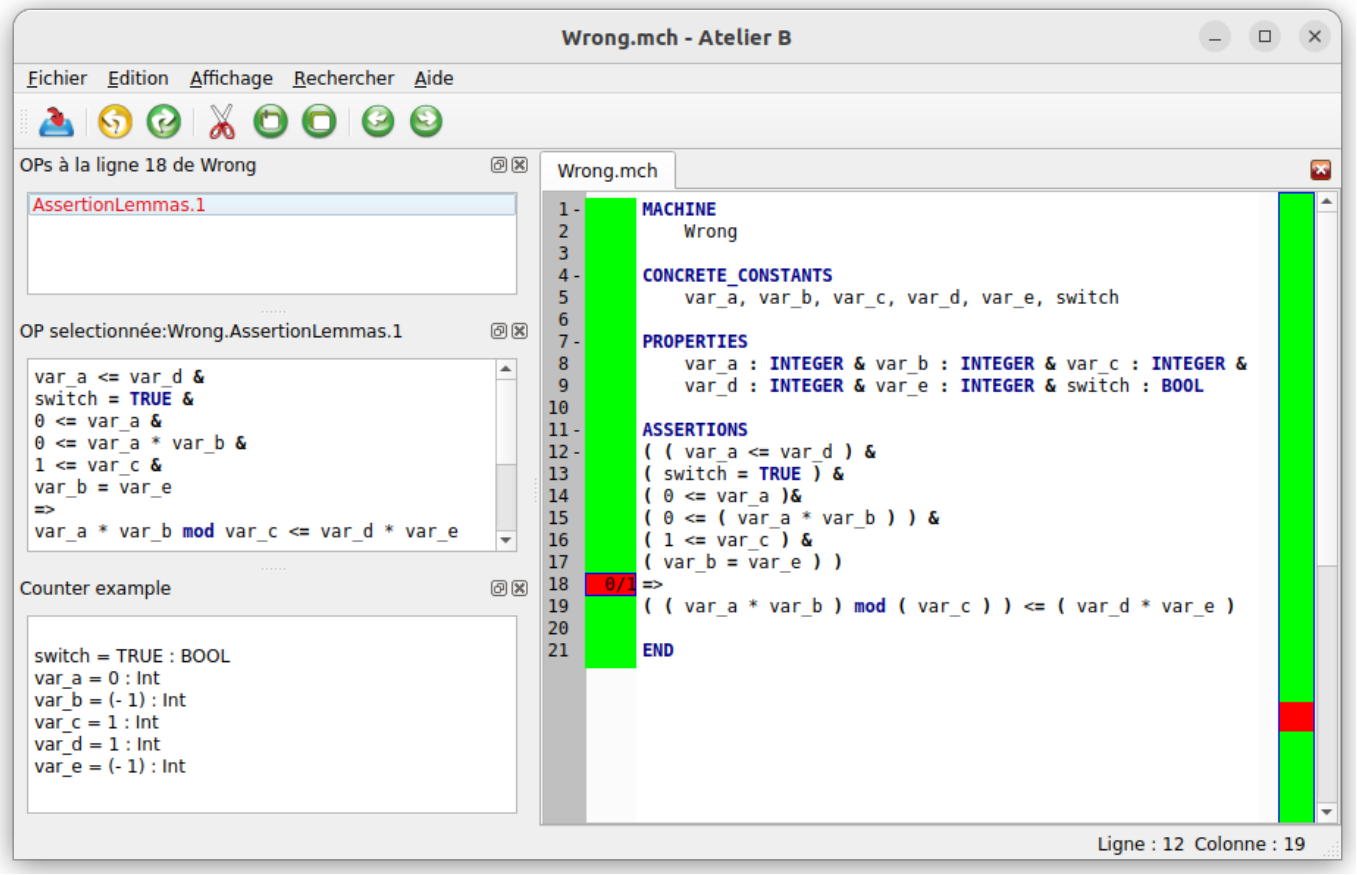

Fig. 3. A simple counter-example in the edition window

6 Limitations

Atelier B currently provides only writers from B to SMT-LIB, so it can only call solvers compatible with the SMT-LIB format.

Not all SMT solvers have the capability to return counter-examples when their result is "sat" and/or "unknown".

An SMT-solver can only process first order logic problems. Some can manage quantified expressions. SMT-LIB has evolved to express finite sets, but most solvers do not implement this logic. For now, the sets are endcoded by ppTransSmt with a membership function that make explicit the properties of sets and functions of B.

Currently, the translator is only included in the mechanisms of cvc4 [3] and z3 [10] as they are the only external solvers with built-in drivers in Atelier B. Implementing a new SMT solver would be as easy as writing its driver. But each

solver has its own way to describe sets, so the translator should be updated as well.

7 Conclusions and future work

This paper presents an ongoing work to use SMT solver to their fullest and reduce the cost of discharging proof obligations by not only sorting them before trying to prove them but also providing information on unachievable proof obligation. This work is embodied in counter_example_reader, a prototype for an additional tool of Atelier B to facilitate comprehension of problems in components.

We plan to improve the prototype by testing it on a representative set of industrial projects. As in a real project can have a lot of variables, an evolution to choose which variables will be shown in counter-examples is already discussed.

The development of a new writer to express sets and B function in SMT-LIB format is considered. It will be compatible only with cvc4 and cvc5 for now but other solvers may follow. counter_example_reader will also be extend to sets and B functions.

Results produced by counter_example_reader are not to be certified as it relies on uncertified solvers, but can still be use in the early stages of projects to direct the attention of engineers on problematic cases.

Acknowledgement

This work was supported by ClearSy System Engineering. Thanks to David Deharbe, engineer at Clearsy, for his supervison.

References

1. Abrial, J.R.: The B-book - assigning programs to meanings (1996)
2. Barrett, C., Fontaine, P., Tinelli, C.: The SMT-LIB Standard: Version 2.6. Tech. rep., Department of Computer Science, The University of Iowa (2017), available at `www.SMT-LIB.org`
3. Barrett, C.W., Conway, C.L., Deters, M., Hadarean, L., Jovanovic, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6806, pp. 171–177. Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_14, bluehttps://doi.org/10.1007/978-3-642-22110-1_14
4. Bobot, F., Filliâtre, J.C., Marché, C., Paskevich, A.: Why3: Shepherd Your Herd of Provers (2011)
5. Déharbe, D., Fontaine, P., Guyot, Y., Voisin, L.: SMT Solvers for Rodin. In: Derrick, J., Fitzgerald, J., Gnesi, S., Khurshid, S., Leuschel, M., Reeves, S., Riccobene, E. (eds.) Abstract State Machines, Alloy, B, VDM, and Z. pp. 194–207. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

6. Delahaye, D., Dubois, C., Marché, C., Mentré, D.: The BWare Project: Building a Proof Platform for the Automated Verification of B Proof Obligations. In: Ait Ameur, Y., Schewe, K.D. (eds.) Abstract State Machines, Alloy, B, VDM, and Z (ABZ). Lecture Notes in Computer Science (LNCS), Springer, Toulouse (France) (Jun 2014), to appear
7. Falampin, J., Le-Dang, H., Leuschel, M., Mokrani, M., Plagge, D.: Improving Railway Data Validation with ProB, pp. 27–43. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-33170-1$_4$, bluehttps://doi.org/10.1007/978-3-642-33170-1_4
8. Leuschel, M., Butler, M.: ProB: A Model Checker for B. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003: Formal Methods. pp. 855–874. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
9. Mentré, D., Marché, C., Filliâtre, J.C., Asuka, M.: Discharging Proof Obligations from Atelier B using Multiple Automated Provers. In: Reeves, S., Riccobene, E. (eds.) ABZ - 3rd International Conference on Abstract State Machines, Alloy, B and Z. Springer, Pisa, Italy (Jun 2012), bluehttps://hal.inria.fr/hal-00681781
10. de Moura, L.M., Bjørner, N.: Z3: An Efficient SMT Solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS. Lecture Notes in Computer Science, vol. 4963, pp. 337–340. Springer (2008), bluehttp://dblp.uni-trier.de/db/conf/tacas/tacas2008.html#MouraB08