Integrating SMT-solvers in Z and B Tools

A. C. Gurgel*, V. G. Medeiros Jr., M. V. M. Oliveira and D. B. P. Déharbe

Departamento de Informática e Matemática Aplicada, UFRN, Brazil

An important frequent task in both Z [14] and B [1] is the proof of verification conditions (VCs). In Z and B, VCs can be predicates to be discharged as a result of refinement steps, some safety properties (preconditions) or domain checking. Ideally, a tool that supports any Z and B technique should among other tasks, automatically discharge as many VCs as possible. Here, we present ZB2SMT [1], a `Java` package designed to clearly and directly integrate both Z and B tools to the satisfiability module theory (SMT) solvers such as veriT [3], a first-order logic (FOL) theorem prover that accepts the SMT syntax [12] as input. By having the SMT syntax as target we are able to easily integrate with further eleven automatic theorem provers that are also compatible like [5,2,7].

veriT provides an open framework to generate certifiable proofs, having a decent efficiency [3] that does not compromise the performance of the tools in usual developments. Its input format is the SMT-LIB language extended with macro definitions. This syntactic facility uses lambda notation and is particularly useful to write formulas containing simple set constructions. This feature enhances the ability of veriT to handle sets, making the solver an interesting tool in formal development efforts in set-based modelling languages.

This prover is used by Batcave [9], an open source tool that generates VCs for the B method. Batcave has a friendly graphic interface and supports B specification with representation in XML format. It uses a parser from the JBTools [13] that is composed by the B Object Library (BOL).

CRefine [11] is a tool that supports the use of the *Circus* refinement calculus. Circus [4] is a concurrent language tailored for refinement that combines Z with CSP [6] and the refinement calculus[10]. CRefine allows the automatic application of refinement laws and discharge of VCs. Much of the VCs generated to validate the refinement law applications, are based on FOL predicates. Hence, CRefine uses veriT to automatically prove such predicates.

In order to allow reuse, we have developed the package ZB2SMT, which integrates elements of Z and B predicates in a common language and transforms these predicates into SMT syntax. ZB2SMT uses an extension of BOL to represent B predicates. On the other hand, the package uses a framework provided by the Community Z Tools (CZT) [8], an ongoing effort that implements tools for standard Z, to represent Z predicates.

In ZB2SMT, Z predicates are converted to B predicates, using the extension of BOL. The extension is needed due to the fact that there are Z operators that do not exist in BOL like the symmetric difference operator. Extending BOL by including these missing operators, we improve the set of predicates that can be treated by ZB2SMT. These predicates are translated into a SMT syntax and written to a file that contains the predicate and some elements such as types of variables, operators definition and set properties that are described over the macro feature. The SMT file is sent to veriT

* The ANP supports the work of the author through the prh22 project.

[1] Freely available at `http://www.consiste.dimap.ufrn.br/projetos/zb2smt`.

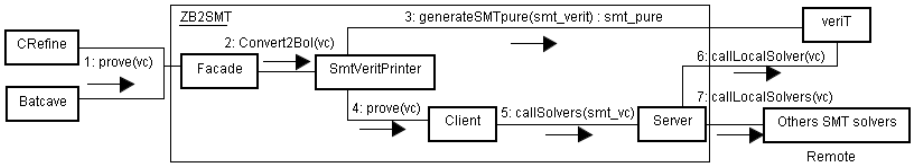

Fig. 1. Collaboration Diagram of ZB2SMT elements

which yields a boolean value for the predicate. On successful evaluation of the predicate, the resulting value is returned. If, however, the evaluation is not successful, veriT can be used to return a SMT file that may be sent to others SMT solvers. This kind of file is a bit different from the original input of veriT since others SMT solvers do not have the peculiarity of macros definition. ZB2SMT allows an integration with the others SMT provers using the conversion from SMT-verit to SMT-pure by veriT.

The application of formal development to large programs generally produces a great amount of VCs. Perform an automatic proof module in only one processor may be impracticable. In order to improve the performance of the proof system, the ZB2SMT has a module that can call different instances of theorem provers on different computers, using socket and `Java`'s thread. The flow of execution of the module in ZB2SMT is illustrated in Figure 1. For conciseness, Figure 1 does not show the conversion from Z to B predicates.

This module has two parts: the client with the information about each possible instance of theorem prover, which can be local or remote, and the server with the theorem prover installed locally. The user creates a configuration for each instance of theorem prover in a file. It contains the following information: path of theorem prover, parameters and the host machine.

The parallelization process replicates VCs and tries to solve by different strategies. Each instance of theorem prover has its own set of specifics strategies and parameters to try to solve the VCs. Thus, the user can create and explore different strategies possibly making the proof process more efficient.

The motivation for our work is to provide a direct verification engine to discharge VCs from Z, B or extensions of their tools. ZB2SMT has been effective and promising in the first experiences in CRefine and Batcave. It can be directly used, like a black box, by tools that work with the CZT framework for Z or B tools which use the BOL library. Furthermore, ZB2SMT offers an easy way to get SMT files from B or Z predicates. Currently, we are embedding, by adjusting parameters and path configurations, others SMT solvers in ZB2SMT.

The parallelism inside ZB2SMT has been an important feature. It improves the proof process by allowing different strategies to be performed in parallel, reducing the verification time. However, the performance of our system can be improved even more by incorporating a predicate classifier. It would classify the predicate and select the best available SMT solver to prove it, since some SMT solvers are more efficient in certain types of predicates.

Acknowledgments. This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES, www.ines.org.br), funded by CNPq grant 573964/2008-4 and by CNPq grant 553597/2008-6.

References

1. J. R. Abrial. *The B Book: Assigning Programs to Meanings.*, volume 1 of *1*. Cambridge University Press, United States of America, 1 edition, 1996.
2. Clark Barrett and Cesare Tinelli. CVC3. In Werner Damm and Holger Hermanns, editors, *Proceedings of the* 19^{th} *International Conference on Computer Aided Verification (CAV '07)*, volume 4590 of *Lecture Notes in Computer Science*, pages 298–302. Springer-Verlag, July 2007. Berlin, Germany.
3. Thomas Bouton, Diego Caminha B. de Oliveira, David Déharbe, and Pascal Fontaine. veriT: An open, trustable and efficient SMT-solver. In *CADE-22 (Int'l Conf. Automated Deduction*, pages 151–156, 2009.
4. A. L. C. Cavalcanti, A. C. A. Sampaio, and J. C. P. Woodcock. A refinement strategy for *Circus*. *Formal Aspects of Computing*, **15**(2–3):146–181, 2003.
5. Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. pages 337–340. 2008.
6. C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.
7. Susmit Jha, Rhishikesh Limaye, and Sanjit Seshia. Beaver: Engineering an efficient smt solver for bit-vector arithmetic. In *Computer Aided Verification*, pages 668–674. 2009.
8. P. Malik and M. Utting. CZT: A Framework for Z Tools. In H. Treharne, S. King, M. C. Henson, and S. A. Schneider, editors, *ZB*, volume **3455** of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2005.
9. E. S. Marinho, V. G. Medeiros Jr, Cláudia Tavares, and David Déharbe. Um ambiente de verificação automática para o método B. In A. C. V. Melo and A. Moreira, editors, *SBMF 2007: Brazilian Symposium on Formal Methods*, 2007.
10. C. Morgan. *Programming from Specifications.* Prentice-Hall, 1994.
11. M. V. M. Oliveira, A. C. Gurgel, and C. G. de Castro. CRefine: Support for the *Circus* Refinement Calculus. In Antonio Cerone and Stefan Gruner, editors, *6th IEEE International Conferences on Software Engineering and Formal Methods*, pages 281–290. IEEE Computer Society Press, 2008. IEEE Computer Society Press.
12. Silvio Ranise and Cesare Tinelli. The SMT-LIB Standard: Version 1.2, 2006. Available at `www.SMT-LIB.org`.
13. J. C. Voisinet. Jbtools: an experimental platform for the formal b method. In *PPPJ '02/IRE '02: Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines, 2002*, pages 137–139, Maynooth, County Kildare, Ireland, Ireland, 2002. National University of Ireland.
14. J. C. P. Woodcock and J. Davies. *Using Z—Specification, Refinement, and Proof.* Prentice-Hall, 1996.