Formal Semantic Foundations for Higher-Order SMT Encoding of B Proof Obligations*

Vincent Trélat[0009−0006−4143−3939]

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
vincent.trelat@inria.fr

1 Introduction and Problem Statement

The B Method [1] is a widely adopted formal method for specifying and verifying safety-critical systems, particularly in the railway industry. Central to its tool chain is the automatic generation of proof obligations (PO) [2], which ensure the consistency of specifications and the preservation of invariants across refinements. Although industrial proof obligation generators routinely produce hundreds of thousands of verification conditions, a significant proportion of these still require interactive proofs. Automation through SMT solvers has long been seen as a promising way to reduce this cost, yet existing approaches are hampered by the limitations of first-order encodings.

In earlier translations to SMT-LIB such as *ppTransSMT* [3], sets were encoded only indirectly, using membership predicates, and functions were represented as relations with extra properties to enforce functionality. The goal of this work is to leverage recent advances in SMT technology—especially the support for higher-order formulas—to provide a more direct, faithful, and effective encoding of B proof obligations.

In previous work, we proposed a practical solution to this problem by introducing a higher-order translation of B proof obligations into SMT-LIB v2.7 [6] and evaluating it experimentally on a large corpus of industrial obligations. That study demonstrated that the new higher-order features of SMT-LIB could be exploited in practice to discharge obligations that had resisted earlier approaches. The emphasis was on the pragmatic side: how to engineer a translator, how to make use of the solver's capabilities, and how much improvement in automation could be achieved. This led to the development of **BEer** (**B E**ncod**er**), a tool implemented in Lean [5] for encoding B proof obligations.

What is still missing, however, is a semantic account of how B proof obligations are to be understood in SMT-LIB, ensuring that the translation is not only efficient but also faithful. This talk proposal focuses precisely on filling this gap. The contribution of this work is twofold. First, it provides a rigorous semantic foundation for the translation of B proof obligations, ensuring that the process is not only pragmatic but also justified from first principles. This contrasts with earlier approaches, which were defined mainly at the level of syntactic encodings and correctness by inspection. Second, by implementing the entire development

* This work is supported by the ANR project BLASST (ANR-21-CE25-0010).

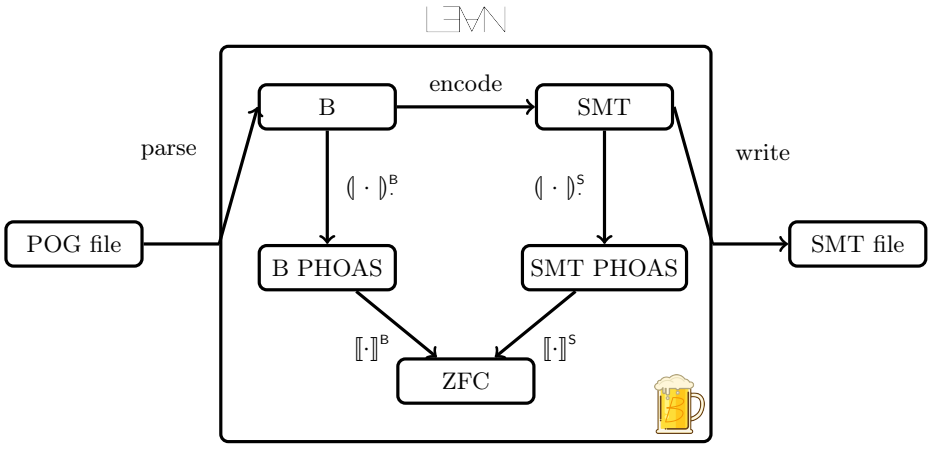

Fig. 1. Architecture of BEer, with the three layers : concrete level (top), abstract level (middle), and semantics (bottom).

in Lean, the work offers a trustworthy and extensible framework in which new language features and encodings can be easily introduced.

2 Semantic Approach

A common approach is to articulate the process through three successive layers: syntax, typing, and semantics. This is illustrated in Figure 1, which depicts the architecture of BEer, clearly separating the concrete, implementation-level layer from the abstract layer, serving as a ground for interpretation and reasoning. At the syntactic level, we distinguish between *concrete terms*, which capture the raw expressions parsed from a proof obligation file, and *abstract terms*, which lift–or *reify*–these expressions into a higher-level representation more suitable for reasoning, commonly called a *Parametric Higher-Order Abstract Syntax* (PHOAS) representation. The abstract layer makes explicit the logical structure of B constructs, providing a bridge between the deeply embedded, unstructured syntax of proof obligations and the typed, well-scoped world of Lean.

Typing then ensures that only well-formed expressions are admitted. In B, typing is implicitly shaped by the mathematical conventions of set theory and well-formedness rules. For a faithful translation it is necessary to make types explicit and to enforce them rigorously. By formalizing the typing system in Lean 4, we obtain a machine-checked guarantee that all terms that reach the encoding stage respect the intended discipline of the language. This provides a safeguard against ill-formed encodings and establishes the foundation for the correctness of the translation.

The semantic layer provides a denotation for B terms within Lean, interpreting terms as a model of ZFC set theory [4] in Lean, thereby capturing the

mathematical meaning of B constructs as intended. This denotational semantics is mirrored on the target side by the higher-order constructs of SMT-LIB v2.7: because SMT-LIB now supports λ-abstractions and higher-order types, it is possible to align the semantics of the source and target more directly than before. The translation is then defined not merely as a syntactic rewriting, but as a semantics-preserving mapping: the meaning of a B term, as given in Lean, coincides with the meaning of its translation, as given in SMT-LIB.

3 Progress and Future Work

This framework has been implemented in Lean: the syntax and typing of a subset of the B and SMT-LIB languages–downsized to core constructs needed for encoding proof obligations–have been formalized, abstract terms have been introduced as an intermediate layer, and a denotational semantics has been defined. This makes it possible to reason formally about B proof obligations before translation, and to establish a clear correspondence with their images in SMT-LIB. While the focus to date has been on the foundational aspects, this semantic basis will in turn guide future extensions to richer fragments of B, such as rational arithmetic and structures, and will underpin ongoing efforts to integrate the encoder into industrial tool chains.

Ongoing work focuses on proving the correctness of the encoding. Initial lemmas have already been established in Lean. The development is however deliberately incremental: the implementation of the encoder involves monadic effects, and reasoning about its correctness requires identifying and maintaining invariants across the sequencing of effectful translation steps, e.g. generation of fresh identifiers, accumulation of context. Formalizing and generalizing these invariants is a tedious effort; completing the full soundness argument therefore remains part of future work.

References

1. Abrial, J.R., Lee, M.K.O., Neilson, D.S., Scharbach, P.N., Sørensen, I.H.: The B-method. In: Prehn, S., Toetenel, H. (eds.) VDM '91 Formal Software Development Methods. pp. 398–405. Springer Berlin Heidelberg, Berlin, Heidelberg (1991)
2. Abrial, J.R., Butler, M., Hallerstede, S., Hoang, T.S., Mehta, F., Voisin, L.: Rodin: an open toolset for modelling and reasoning in Event-B. International Journal on Software Tools for Technology Transfer **12**(6), 447–466 (Nov 2010). https://doi.org/10.1007/s10009-010-0145-y
3. Déharbe, D.: Integration of SMT-solvers in B and Event-B development environments. Science of Computer Programming **78**, 310–326 (03 2013). https://doi.org/10.1016/j.scico.2011.03.007
4. Fraenkel, A.A., Bar-Hillel, Y.: Foundations of Set Theory. Elsevier, Atlantic Highlands, NJ, USA (1973)
5. de Moura, L., Kong, S., Avigad, J., van Doorn, F., von Raumer, J.: The Lean Theorem Prover (System Description). In: Felty, A.P., Middeldorp, A. (eds.) Automated Deduction - CADE-25. pp. 378–388. Springer International Publishing, Cham (2015)

6. Trélat, V.: Safely Encoding B Proof Obligations in SMT-LIB. In: Leuschel, M., Ishikawa, F. (eds.) Rigorous State-Based Methods. pp. 52–69. Springer Nature Switzerland, Cham (2026)