UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Tomaz Gomes Mascarenhas

Demonstrando teoremas em Lean por meio da reconstrução de provas em SMT

Belo Horizonte
2023

Tomaz Gomes Mascarenhas

Demonstrando teoremas em Lean por meio da reconstrução de provas em SMT

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Haniel Barbosa

Belo Horizonte
2023

Tomaz Gomes Mascarenhas

Proving Lean theorems via reconstructed SMT proofs

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Haniel Barbosa

Belo Horizonte
2023

[Ficha Catalográfica em formato PDF]

A ficha catalográfica será fornecida pela biblioteca. Ela deve estar em formato PDF e deve ser passada como argumento do comando `ppgccufmg` no arquivo principal `.tex`, conforme o exemplo abaixo:

```
\ppgccufmg{
    ...
    fichacatalografica={ficha.pdf}
}
```

[Folha de Aprovação em formato PDF]

A folha de aprovação deve estar em formato PDF e deve ser passada como argumento do comando `ppgccufmg` no arquivo principal `.tex`, conforme o exemplo abaixo:

```
\ppgccufmg{
      ...
      folhadeaprovacao={folha.pdf}
}
```

Dedicuum cest laborae a quelquis personatum que ajudorat a facirelo.

Acknowledgments

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.

“*Só quem sonha acordado vê o sol nascer.*”

(Unknown)

Resumo

Apesar de sua expressividade e robustez, assistentes de demonstração podem ser proibitivamente custosos para serem usados em formalizações de grande escala, dada a dificuldade de produzir as demonstrações interativamente. Atribuir a responsabilidade de demonstrar algumas das proposições a provadores automáticos de teoremas, como solucionadores de satisfatibilidade modulo teorias (SMT), é um jeito reconhecido de melhorar a usabilidade de assistentes de demonstração. Essa dissertação descreve uma nova integração entre o assistente de demonstração Lean 4 e o solucionador SMT cvc5.

Dada uma codificação de um teorema declarado em Lean como um problema de SMT e uma demonstração provida pelo cvc5 para o problema codificado, nós mostramos como traduzir essa demonstração para uma que certifique o teorema original em Lean. Para isso é necessário demonstrar a corretude, em Lean, dos passos lógicos tomados pelo solucionador. Desse modo, o verificador de demonstrações de Lean aceitará a demonstração em SMT do teorema original, caso o processo seja bem sucedido.

Essa ferramenta é parte do projeto em conjunto Lean-SMT, que tem como objetivo criar uma tática em Lean que implemente o processo completo, isto é, a pártir de um teorema em Lean, traduzi-lo para um problema formulado na linguagem SMT-Lib, invocar um solucinador SMT para tentar resolvê-lo e produzir uma demonstração, e, caso ele seja bem-sucedido, traduzi-la para certificar o teorema original em Lean (o que é feito por nossa ferramenta). Todas as etapas desse processo estão em estado avançado de desenvolvimento.

Palavras-chave: Verificação Formal, Lean, SMT

Abstract

Despite their expressivity and robustness, interactive theorem provers (ITPs) can be prohibitively costly to use in large-scale formalizations due to the burden of interactively proving goals. Discharging some of these goals via automatic theorem provers, such as satisfiability modulo theories (SMT) solvers, is a known way of improving the usability of ITPs. This thesis describes a novel integration between the ITP Lean 4 and the SMT solver cvc5.

Given the encoding of some Lean goal as an SMT problem and a proof from cvc5 of the encoded problem, we show how to lift this proof into a proof of the original goal. This requires proving the correctness, inside Lean, of the steps taken by the solver. Thus Lean's proof checker will accept the SMT proof as a proof of the original goal, in case this process is successful.

This tool is part of the joint project Lean-SMT, which aims to create a tactic in Lean that implements the whole pipeline, that is, from a goal in Lean, translate it into a query in SMT-Lib format, try to prove it using a SMT solver and, in case it is successful, lift the proof produced, closing the original goal in Lean (which is done by our tool). All the steps of the pipeline are in an advanced stage of development.

Keywords: Formal Verification, Lean, SMT

Contents

Chapter 1

Introduction

1.1 Context

The process of generating mechanized proofs, for example for the correctness of a given program according to a specification, can be divided into two categories: interactive and automatic.

Interactive theorem provers (ITPs) are mainly represented by proof assistants, in which, after defining a theorem, the user attempts to manually write a proof for it, relying on the tool to organize the set of hypothesis and how the goal changed step-wisely through the proof, as well as to ensure the correctness of each step according to a small, trusted kernel. Each logic step must be explicitly stated by the user, which makes the tool costly to be used.

Automatic theorem provers (ATPs), on the other hand, only require the user to define a conjecture, proceeding automatically to determine whether there exists a proof for it, or possibly providing a counter-example if there is one. For our setting, the most relevant kind of ATP are the satisfiability modulo theories (SMT)[5] solvers. Although they are easier to use, ATPs require a large codebase to implement all the algorithms necessary to execute the search for a proof, making them more susceptible to errors and harder to be trusted, since the larger is the codebase, the more complicated it is to verify it, and, also, once it is verified, it's development becomes freezed (otherwise it would have to be verified again).

A common approach to address the trust issue for ATPs is to have them provide a proof to support their results, so that it can be independently verified whether it indeed proves the theorem in question. Via these proofs the automatic proving performed by ATPs can be leveraged by ITPs, since their requirement to accepting a proof, i.e. that each step is correct according to its internal logic, can be applied to the ATP proof. By connecting these systems, the user could have all the freedom to use its own creativity and expertise in writting proofs that the ITPs offer, while delegating the burden of proofs that are long and monotonous to ATPs. Indeed, this connection is so important that there are projects like Hammering Towards QED [7] that outline all the efforts that were already made in order to integrate interactive and automatic theorem provers.

1.2 Related Work

1.2.1 SMTCoq

One notable example of such integrations is SMTCoq [13]. It is a plugin for the proof assistant Coq [6] that can be used as a tactic to prove theorems via their encoding into SMT and by lifting proofs produced by the SMT solvers veriT [10] and CVC4 [3]. The tool relies on a preprocessor written in OCaml to transform proof witnesses coming from different solvers into certificates in the Coq language. The system has a set of checkers for each theory in SMT, each one of them consisting of theorems asserting the validity of certain transformations in the SMT terms. All those checkers are connected by the main checker, that is essentially a theorem stating that if all the transformations resulted in an empty clause, then the lifting of the original term is false, for any instantiation of its free variables. This kind of reasoning is known as proof by computational reflection [9] which is an instance of Certified Transformations, which will be described in Section 3.1.

1.2.2 Sledgehammer

The ITP Isabelle/HOL [15] has a similar tool, namely, Sledgehammer [8]. This system achieves its goal by invoking several SMT solvers in parallel to prove a given goal and collecting their output to determine which lemmas must be applied in order to prove the theorem inside Isabelle. In a way, this approach is very similar to the one we're using in this project, as the proof is produced on the fly (known as the Certifying approach, which will also be described in Section 3.1) as opposed to having a single theorem that establishes once and for all that, if all steps performed by the solver were successful, then the original goal is valid, as is done by SMTCoq.

1.2.3 Hammering Towards QED

As previously mentioned, Hammering Towards QED is a project that aims to describe all the tools, which the paper calls “hammers”, that were created with the purpose of connecting automatic and interactive theorem provers. Besides that, this document also outlines the main components that such tools usually have. They are the following:

- The premiss selector, that is a module that identifies a subset of the facts previously demonstrated in the ITP that are more likely to be useful in order to prove the given goal, to be dispatched to the ATP.
- The translation module, that builds a problem in the language of the ATP that corresponds to the original goal from the ITP.
- The proof reconstruction module, that lifts the proof produced by the ATP into a proof that is accepted by the ITP.

Moreover, the main strategies used to reconstruct the proof produced by the automatic system inside the interactive one are also reported. We give a brief description of them:

- Parsing each step of the proof into predefined lemmas or tactics from the ITP and replay them inside the system.
- Use the ITP to verify a deeply embedded version of the proof received, and, in case it is succesful, reflect this proof inside it’s checker to prove the original goal.
- Compile the proof into the ITP’s source code. This implies generating an actual script in the native language of the interactive system that corresponds to the proof received. This method, as opposed to the previous two, has the advantage of not requiring access to the ATP every time the proof is checked, but only on the first time.

1.3 Contributions

Given this context, we present a tool that would be an essential part of the integration between the ITP Lean 4 [12] and the SMT solver cvc5 [1]. Specifically, we aim to build a system that takes proofs of the unsatisfiability of SMT queries produced by cvc5 and reconstructs and checks them using Lean. The main motivation of this project is that despite the fact that Lean is emerging as a promising programming language and proof assistant and being widely used by mathematicians in large-scale formalizations [14, 11], there is currently no way to interact with SMT solvers from it, even though these systems have been central in previous developments of proof automation in ITPs, as seen in Sections 1.2.1 and 1.2.2. The contribution of the present work would enable a faster development of this kind of project using Lean.

We use the cvc5 solver because it already has a module for exporting proofs as Lean scripts [2], using a representation of the SMT terms[1] as an inductive type in Lean.

Note that, as opposed to SMTCoq and Sledgehammer that implements all the three modules of a hammer (as described in Section 1.2.3), our system only implements the third module for now, despite the end goal of this project being to implement the complete integration. Also, the reconstruction technique that we use is the first one listed in Section 1.2.3, as we will describe in the next sections.

1.4 Organization of this document

[1]For more details about the SMT term language, see SMT-LIB [4].

Chapter 2

Formal Preliminaries

2.1 Satisfiability Modulo Theories

2.2 Lean's Type Theory

- Falar sobre porque eh facil confiar no proof assistant
- Explicar que taticas extendem a linguagem mas nao aumentam o trusted core

2.3 Lean's Framework for Metaprogramming

Chapter 3

Certifying Reconstruction of SMT Proofs in Lean

3.1 Certified vs Certifying

3.2 Classical vs Intuitionist (?)

3.3 Tactics

3.4 The Complete Architecture

3.5 Skipping the Parser

Chapter 4

Evaluation

Chapter 5

Future Work

Bibliography

[1] Haniel Barbosa et al. cvc5: A versatile and industrial-strength smt solver. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 415–442, Cham, 2022. Springer International Publishing.

[2] Haniel Barbosa, Andrew Reynolds, Gereon Kremer, Hanna Lachnitt, Aina Niemetz, Andres Notzli, Alex Ozdemir, Mathias Preiner, Arjun Viswanathan, Scott Viteri, Yoni Zohar, Cesare Tinelli, and Clark Barrett. Flexible proof production in an industrial-strength smt solver. In Jasmin Blanchette, Laura Kovacs, and Dirk Pattinson, editors, *International Joint Conference on Automated Reasoning (IJCAR)*, Lecture Notes in Computer Science. Springer, 2022.

[3] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification (CAV)*, pages 171–177. Springer, 2011.

[4] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa, 2017. Available at www.SMT-LIB.org.

[5] Clark W. Barrett and Cesare Tinelli. Satisfiability modulo theories. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking.*, pages 305–343. Springer, 2018.

[6] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[7] Jasmin C. Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards qed. *Journal of Formalized Reasoning*, 9(1):101–148, Jan. 2016.

[8] Jasmin Christian Blanchette, Sascha Böhme, and Lawrence C. Paulson. Extending sledgehammer with smt solvers. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction – CADE-23*, pages 116–130, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[9] Samuel Boutin. Using reflection to build efficient and certified decision procedures. In Martín Abadi and Takayasu Ito, editors, *Theoretical Aspects of Computer Software*, pages 515–529, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[10] Thomas Bouton, Diego Caminha B. de Oliveira, David Déharbe, and Pascal Fontaine. veriT: An Open, Trustable and Efficient SMT-Solver. In Renate A. Schmidt, editor, *Proc. Conference on Automated Deduction (CADE)*, volume 5663 of *Lecture Notes in Computer Science*, pages 151–156. Springer, 2009.

[11] Davide Castelvecchi. Mathematicians welcome computer-assisted proof in "grand unification" theory. *Nature*, 595, 06 2021.

[12] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25*, pages 378–388, Cham, 2015. Springer International Publishing.

[13] Burak Ekici, Alain Mebsout, Cesare Tinelli, Chantal Keller, Guy Katz, Andrew Reynolds, and Clark Barrett. Smtcoq: A plug-in for integrating smt solvers into coq. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 126–133, Cham, 2017. Springer International Publishing.

[14] The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, pages 367–381, New York, NY, USA, 2020. Association for Computing Machinery.

[15] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.