An Interactive SMT Tactic in Coq using Abductive Reasoning

Arjun Viswanathan
Dept. of CS
University of Iowa
Iowa City, USA
arjun.viswanathan23@gmail.com

Chantal Keller
Laboratoire Méthodes Formelles
Université Paris-Saclay
Orsay, France
Chantal.Keller@lri.fr

Cesare Tinelli
Dept. of CS
University of Iowa
Iowa City, USA
cesare-tinelli@uiowa.edu

***Abstract*—Interactive theorem provers (ITPs) and automatic theorem provers (ATPs) offer very different functionalities in hardware and software verification. ITPs provide high assurances and low automation, while ATPs offer push-button proofs but are less reliable. SMTCoq is a tool that allows a user of the Coq ITP to delegate parts of their proof to an SMT solver (an ATP), without losing the proof guarantees of the Coq kernel. It does this via a tactic, which tries to dispatch the current goal to an SMT solver. If the solver finds a counterexample to the goal, it returns this to the Coq user. We enhance this integration with the SMT solver, allowing it instead to ask for facts that would allow it to prove the goal, via abductive reasoning.**

***Index Terms*—abduction, cvc5, SMTCoq, Coq, SMT**

I. Introduction

Interactive theorem provers (ITPs) [1] are used to prove logical properties in various hardware and software verification tasks. Interactive proofs are rigorous, but tedious, involving elaborate sub-proofs of even simple facts that could be automatically solved by automatic theorem provers (ATPs) such as SMT (satisfiability modulo theories) [2] solvers. Due to its proof approach, adding an SMT (satisfiability modulo theories) solver to the trust-base of an ITP is a liability to its trustworthiness. SMTCoq [3] is an integration that achieves the best of both worlds for certain logic fragments. Using one of SMTCoq's tactics, a Coq [4] user is able to dispatch goals to a SAT/SMT solver without expanding Coq's trust-base. SMTCoq achieves this by taking from the SMT solver, along with its result, a certificate, and using this certificate to construct a proof term of the corresponding formula within Coq's logic.

The traditional integration between Coq and an SMT solver, say cvc5 [5], via SMTCoq is as follows. The user calls the `cvc5` tactic - provided by SMTCoq in Coq - on a goal G, which asks cvc5 to prove G. If cvc5 succeeds, it also sends a proof certificate, which SMTCoq uses to derive a proof in Coq's logic using a process called computational reflection. If cvc5 fails because it found a counterexample disproving G, it presents this to the user, and the user is left to ponder whether the counterexample was derived because the solver didn't have enough information, or if G is an invalid statement.

For cases where the solver finds a counterexample, we introduce the `cvc5_abduct` tactic which asks the SMT solver for a formula that would entail the goal G. With this tactic, we imagine a more interactive session between the Coq user and the SMT solver, where the solver might be able to answer the question of what information it is lacking, in order to prove the goal.

II. Abduction via SyGuS

The cvc5 SMT solver performs abductive reasoning [6] via syntax-guided synthesis (SyGuS) [7]. Given a set of formulas H (taken as a conjunction), and a goal G, an abduct is a formula ϕ such that $H \wedge \phi \models G$. Furthermore, cvc5 finds abducts modulo some theory T, so that the entailment holds in T, and that the abduct is consistent with the axioms in T. Internally, since cvc5 uses SyGus to find abducts, the search-space for the abduct is restricted syntactically by a grammar R, that may be produced as a parameter to the abduction solver.

III. The `cvc5_abduct` Tactic

The `cvc5_abduct` tactic takes a positive integer parameter that indicates the number of abducts the user wants from cvc5. The disjunction of the abducts entails the goal, which is to say that each abduct individually suffices to entail the goal.

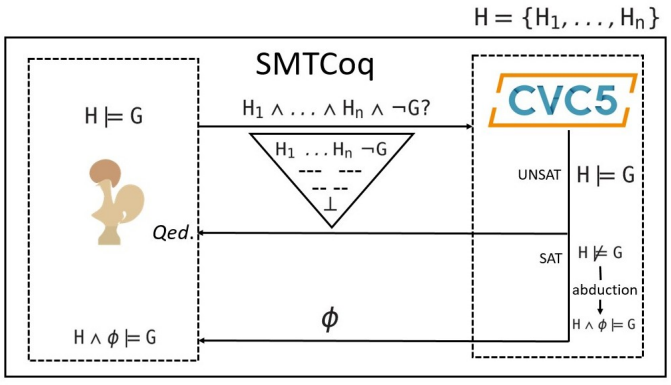

Fig. 1. Interactions between Coq and cvc5 in SMTCoq.

Fig. 1 illustrates the enhancement to the Coq-cvc5 integration that abduction provides. Notice that a proof goal in Coq turns to an unsatisfiability goal in cvc5. The proof certificate returned by cvc5 is a tree that derives the empty clause ($\bot$) from the negation of the input, proving its unsatisfiability.

In cases where the SMT solver finds the negation of the goal to be satisfiable, our tactic provides an alternative to counterexamples, where cvc5 returns an abduct ϕ.

In the following, we present an example of how this enhancement may be used. Suppose our Coq development contains a binary function `f` of type `Z → Z → Z` (where `Z` is Coq's integer type), and many facts about `f`. We use the `cvc5` tactic to ask the SMT solver to prove the following goal about `f`:

```
forall (x y z : Z), x = y + 1
       →(f y z) = f z (x − 1).
```

The solver returns the following counter-example:

```
f = λ x, y → x
x = 1, y = 0, z = 1
```

Instead of trying to determines the facts about f that would eliminate this and future counter-examples, the user may invoke the abduction solver via `cvc5_abduct 3` tactic, which presents 3 abducts:

```
z = y
z + 1 = x
f z y = f y z
```

The third abduct might suggest to the user that cvc5 only needs to know that the function is commutative to prove the goal, and a subsequent call to the `cvc5` tactic, with a proof of the commutativity of `f` (the SMTCoq tactics allow for hypotheses to be passed as arguments) would successfully close the proof.

IV. Conclusion and Future Work

We have extended SMTCoq by adding to its traditional set of proof tactics, a more interactive tactic called `cvc5_abduct`. When cvc5 finds a goal to be invalid, this tactic presents an alternative to presenting counterexamples — it relies on the abductive capabilities of cvc5 to present fact(s) that would entail the goal, to the user. With tools that deal with integrating ATPs and ITPs such as hammers [8] [9], a good premise selection strategy is important to avoid either overloading the ATP with too many facts, or conversely supplying it with insufficient facts to solve the goal [10] [11]. With the abduction tactic, we imagine the ATP as being part of this premise selection process.

The development of the `cvc5_abduct` tactic is still under development. There is a version that works in a branch, but only under certain restrictive parameters. Our first goal in this project is to lift these restrictions and have a working version of the tactic that is available in an SMTCoq release. Beyond that, there are many ways in which the interaction with the abduction solver could be improved. Currently, we don't modify the default grammar that cvc5 selects to produce abducts, one that contains rules for the generating the entire language under consideration. A combination of allowing grammar selection by the user and using automatic methods to reduce the language generated by the grammar would allow for better abducts.

The entire abduction pipeline is implemented manually now, where the Coq user looks at the set of abducts from the solver and uses their understanding of the imported libraries and other available facts and selects any facts that may be implied by an abduct. One can imagine automating this process using a smarter tactic that is able to search the open Coq libraries for facts that match the abducts and automatically complete the proof by including those facts.

References

[1] Y. Bertot and P. Castran, *Interactive Theorem Proving and Program Development: Coq'Art The Calculus of Inductive Constructions*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[2] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking.*, 2018, pp. 305–343. [Online]. Available: https://doi.org/10.1007/978-3-319-10575-8_11

[3] M. Armand, G. Faure, B. Grégoire, C. Keller, L. Théry, and B. Werner, "A modular integration of sat/smt solvers to coq through proof witnesses," in *Proceedings of the First International Conference on Certified Programs and Proofs*, ser. CPP'11. Berlin, Heidelberg: Springer-Verlag, 2011, p. 135–150. [Online]. Available: https://doi.org/10.1007/978-3-642-25379-9_12

[4] L. Théry, P. Letouzey, and G. Gonthier, *Coq*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 28–35. [Online]. Available: https://doi.org/10.1007/11542384_6

[5] H. Barbosa, C. W. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, and Y. Zohar, "cvc5: A versatile and industrial-strength SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, ser. Lecture Notes in Computer Science, D. Fisman and G. Rosu, Eds., vol. 13243. Springer, 2022, pp. 415–442. [Online]. Available: https://doi.org/10.1007/978-3-030-99524-9_24

[6] A. Reynolds, H. Barbosa, D. Larraz, and C. Tinelli, "Scalable algorithms for abduction via enumerative syntax-guided synthesis," in *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, N. Peltier and V. Sofronie-Stokkermans, Eds., vol. 12166. Springer, 2020, pp. 141–160. [Online]. Available: https://doi.org/10.1007/978-3-030-51074-9_9

[7] R. Alur, R. Bodik, G. Juniwal, M. M. K. Martin, M. Raghothaman, S. A. Seshia, R. Singh, A. Solar-Lezama, E. Torlak, and A. Udupa, "Syntax-guided synthesis," in *2013 Formal Methods in Computer-Aided Design*, 2013, pp. 1–8.

[8] J. C. Blanchette, S. Böhme, and L. C. Paulson, "Extending sledgehammer with smt solvers," in *Automated Deduction – CADE-23*, N. Bjørner and V. Sofronie-Stokkermans, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 116–130.

[9] L. Czajka and C. Kaliszyk, "Hammer for coq: Automation for dependent type theory," *Journal of Automated Reasoning*, vol. 61, 06 2018.

[10] J. Alama, T. Heskes, D. Kühlwein, E. Tsivtsivadze, and J. Urban, "Premise selection for mathematics by corpus analysis and kernel methods," *J. Autom. Reason.*, vol. 52, no. 2, pp. 191–213, 2014. [Online]. Available: https://doi.org/10.1007/s10817-013-9286-5

[11] D. Kühlwein, T. van Laarhoven, E. Tsivtsivadze, J. Urban, and T. Heskes, "Overview and evaluation of premise selection techniques for large theory mathematics," in *Automated Reasoning*, B. Gramlich, D. Miller, and U. Sattler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 378–392.