Reflection-based Integration of SMT Solvers into Theorem Provers

by

Yan Peng

B. Engineering, Zhejiang University, 2012

M. Science, University of British Columbia, 2015

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University of British Columbia

(Vancouver)

April 2192

© Yan Peng, 2192

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Reflection-based Integration of SMT Solvers into Theorem Provers

submitted by **Yan Peng** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Computer Science**.

Examining Committee:

Mark R. Greenstreet, Department of Computer Science, University of British Columbia
Supervisor

Alan J. Hu, Department of Computer Science, University of British Columbia
Supervisory Committee Member

Ronald Garcia, Department of Computer Science, University of British Columbia
Supervisory Committee Member

Matt Kaufmann, Department of Computer Sciences, University of Texas at Austin
Supervisory Committee Member

xx, Department
University Examiner

xx, Department
External Examiner

Abstract

SMT solvers are automated decision procedures for domain-specific solving problems including but not limited to boolean, integer, and real theories, uninterpreted function theories, array theories, and bit-vector theories. They have become a main method widely applied in the industry for solving software and hardware formal verification problems. Though theorem provers are largely interactive, they are often resorted to for its powerful induction capability, abstraction capability, and strong proof management support. Any moderate-complexity problem will require both methods but switching between tools requires modelling the same system in each tool. This introduces huge verification overhead and could be erroneous.

Existing work on the integration of SMT solvers and theorem provers focuses on proof reconstruction for soundness, but lacks emphasis on the usability and proof efficiency. To use such integration, the theorem prover term needs to be in a form that is ready for direct translation. On the other hand, since proof reconstruction is undecidable, it is reported that for some problems the lemma could be time consuming and even undecidable. In this thesis, We stress the problem of the usability of such integration. We propose using reflection for integrating SMT solvers into theorem provers. We note that using proof reconstruction to remove the dependency on the soundness of the SMT solver is an orthogonal research direction.

We demonstrate our method using the ACL2 theorem prover and the Z3 SMT solver. To bridge the gap between the untyped first-order logic of ACL2 and the many-sorted logic of SMT solvers, we apply reflection over the ACL2 terms to achieve adding user hypotheses, function expansion, type inference, term replacement and many other transformations. By using reflection, the user is freed from

manually transforming theorem prover terms into a shape that is directly translatable. In addition, soundness of these transformations is easily achieved and does not require extra proof time, therefore promotes proof efficiency. The framework is also built to be extensible, allowing new transformations to be integrated. We analyze our tool by conducting formal verification of asynchronous circuits and machine learning convex optimization problems.

Lay Summary

The lay or public summary explains the key goals and contributions of the research/ scholarly work in terms that can be understood by the general public. It must not exceed 150 words in length.

Preface

At University of British Columbia (UBC), a preface may be required. Be sure to check the Graduate and Postdoctoral Studies (GPS) guidelines as they may have specific content to be included.

Table of Contents

List of Tables

List of Figures

Glossary

This glossary uses the handy `acroynym` package to automatically maintain the glossary. It uses the package's `printonlyused` option to include only those acronyms explicitly referenced in the LaTeX source. To change how the acronyms are rendered, change the `\acsfont` definition in `diss.tex`.

GPS Graduate and Postdoctoral Studies

Acknowledgments

Thank those people who helped you.

Don’t forget your parents or loved ones.

You may wish to acknowledge your funding sources.

Chapter 1

Introduction

1.1 Some Section

some citation[1]

some citation[2]

Chapter 2

Related Work

Chapter 3

Architecture of Smtlink

Chapter 4

Soundness

4.1 The Soundness Argument

Given the original goal G, `Smtlink` uses a series of verified clause-processors to transforms G into a goal G_{tcp} that can be readily translated into `Z3`. Clause processors are programs that transform an input clause into a conjunction of output clauses. When a clause processor is verified, the theorem that the conjunction of output clauses imply the input clause is established. Therefore, we know G_{tcp} implies the original goal G and the transformation steps are sound. After the verified clause-processors, a final trusted clause-processor translates G_{tcp} into `Z3`. We refer to this translated form as G_{smt}.

Let $x_1, x_2, \ldots, x_n$ denote the free variables in G_{tcp}, let G_{smt} denote the translated goal, and $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n$ denote the free variables of G_{smt}.

For soundness, we want

$$\text{SMT} \vdash G_{\text{smt}} \quad \Rightarrow \quad \text{ACL2} \vdash G_{\text{tcp}} \tag{4.1}$$

In the remainder, we assume

- ACL2 and the SMT solver are both sound for their respective theories.
- The SMT solver is a decision procedure for a decidable fragment of first-order logic. In particular, this holds for Z3, the only SMT solver that is cur-

rently supported by `Smtlink`. In addition, we are working with a quantifier-free fragment of Z3's logic.

- There is a one-to-one correspondence between the free variables of G_{tcp} and the free variables of G_{smt}. This is the case with the current implementation of `Smtlink`.

Theorem 4.1.1 *for each model in ACL2, there exists a model in Z3*

Now, suppose that G_{tcp} is not a theorem in ACL2. Then, by Gödel's Completeness Theorem, there exists a model of the ACL2 axioms that satisfies $\neg G_{\text{tcp}}$. We need to show that in this case there exists a model of the SMT solver's axioms that satisfies $\neg G_{\text{smt}}$. There are two issues that we must address. First, we need to provide, for the interpretation of any function symbol $\texttt{f}_{\text{acl2}}$ in G_{tcp}, an interpretation for the corresponding function symbol $\texttt{f}_{\text{smt}}$ in G_{smt}. This brings us to the second issue: the logic of ACL2 is untyped, but the logic of SMT solvers including Z3 is many-sorted. Thus, there are models of the ACL2 axioms that have no correspondence with the models of the SMT solver. We restrict our attention to goals, G_{tcp} where the type of each subterm in the formula can be deduced. We refer to such terms as translatable. If G_{tcp} is not translatable, then `Smtlink` will fail to prove it.

Definition 4.1.1 *translatable's definition*

For the remainder, we restrict our attention to translatable goals. Because G_{tcp} is translatable, there is a set R of unary recognizer functions (primitives such as rationalp that return a boolean) and also a set S of other functions, such that every function symbol in G_{tcp} is a member of R or of S, and every function in S is "well-typed" with respect to R in some sense that we can define roughly as follows. We associate each function symbol $\texttt{f}_{\text{acl2}}$ in S with a function symbol $\texttt{f}_{\text{smt}}$ of Z3, and each predicate r in R with a type in Z3. The trusted clause processor checks that there is a "type-hypothesis" associated with every free variable of G_{tcp} and a fixing function for every type-ambiguous constant (e.g. `nil`) – G_{tcp} holds trivially if any of these type-hypotheses are violated. For every function $\texttt{f}_{\text{acl2}}$ in S, we associate a member of R to each of its arguments (i.e. a "type") and also to the result. For user-defined functions (i.e. uninterpreted function for the SMT solver), `Smtlink`

generates a subgoal for each call to $\texttt{f}_{\text{smt}}$: if the arguments satisfy their declared types (i.e., predicates from R), then the result must satisfy its declared type as well. For built-in ACL2 functions (e.g. `binary-plus`) we assume the "obvious" theorems are present in the ACL2 logical world. Now suppose we have a model, M_1, of $\neg G_{\text{tcp}}$, and consider the submodel, M_2, containing just those objects m such that m satisfies at least one predicate in R that occurs in G_{tcp}. Note that M_2 is closed under (the interpretation of) every operation in S, because $\neg G_{\text{tcp}}$ implies that all of the "type-hypotheses" of G_{tcp} are true in M_1. This essentially excludes "bad atoms", as defined by the function `acl2::bad-atom`. Then because G_{tcp} is quantifier-free, M_2 also satisfies $\neg G_{\text{tcp}}$. We can turn M_2 into a model M_2' for the language of Z3, by assigning the appropriate type to every object. (As noted in Section **??**, M_2' satisfies the theory of Z3 if M_2 is a model of ACL2(r); but for ACL2 that is not the case, so in future work, we expect to construct an extension of M_2' that satisfies all of the axioms for real closed fields.) Then we have the claim: for every assignment s from the free variables of G_{tcp} to M_2 with corresponding typed assignment s' from the free variables of G_{smt} to M_2' , if $\neg G_{\text{tcp}}$ is true in M_2 under s, then $\neg G_{\text{smt}}$ is true in M_2' under s'. Thus, if G_{tcp} is translatable, and $\neg G_{\text{smt}}$ is unsatisfiable, we conclude that G_{tcp} is a theorem in ACL2.

In the rest of this section, we discuss for each of the recognizer functions and each of the basic functions in ACL2, how we associate them with the corresponding Z3 functions

4.2 Soundness for Uninterpreted Functions

4.3 Soundness for Types

4.4 The Trusted Clause Processor

Chapter 5

Formally Verifying the Asynchronous Pipelines

Chapter 6

Formally Verifying the Cauchy-Schwarz Inequality

Chapter 7

Comparing Experiments

Chapter 8

Conclusions

Bibliography

[1] P. J. Cohen. The independence of the continuum hypothesis, 1963. → page 1

[2] A. Einstein. Concerning an heuristic point of view toward the emission and transformation of light. *Annalen Phys*, pages 132–148, 1905. → page 1

Appendix A

Supporting Materials

This would be any supporting material not central to the dissertation. For example:

- additional details of methodology and/or data;
- diagrams of specialized equipment developed.;
- copies of questionnaires and survey instruments.