MetiTarski's Menagerie of Cooperating Systems

Lawrence C. Paulson

Computer Laboratory, University of Cambridge, England
lp15@cl.cam.ac.uk

Abstract. MetiTarski, an automatic theorem prover for real-valued special functions, is briefly introduced. Its architecture is sketched, with a focus on the arithmetic reasoning systems that it invokes. Finally, the paper describes some applications where MetiTarski is itself invoked by other tools.

1 Introduction

As we all know, connecting systems together is easy; the difficulty lies in getting them to cooperate productively. Combining theorem proving with computer algebra has long been regarded as a promising idea, but it has been difficult to realise in practice. MetiTarski is an automatic theorem prover for real-valued special functions [2]. In its original form it consisted of two separate systems linked together: Metis [14, 15] (a resolution theorem prover) and QEPCAD [5, 13] (a quantifier elimination procedure for real-closed fields). Today, MetiTarski can invoke three separate reasoning tools (QEPCAD, Mathematica and Z3) and can itself be invoked by other tools, in particular, KeYmaera and PVS.

2 Architectural Overview

The core idea in MetiTarski is to reduce problems involving special functions (sin, cos, ln, etc.) to decidable polynomial inequalities, which can then be supplied to QEPCAD. First-order formulas over polynomial inequalities over the real numbers admit quantifier elimination [11], and are therefore decidable. This decision problem is known as RCF, for real closed fields. Dolzmann et al. [10] have written a useful overview of both the theory and its practical applications.

An early design decision was to adopt an existing theorem prover (namely Metis), rather than to write a tableau-style theorem prover from scratch, which was the approach adopted for Analytica [7] and Weierstrass [3], two earlier systems that combined mathematical software with logic. It seemed clear to us that the resolution method would turn out to be much more sophisticated and effective than the naive methods our small group would be able to concoct on our own. Instead of having to write an entire theorem prover, we would merely need to write some interface code and modify certain standard aspects of resolution. Arithmetic simplification obviously had to be introduced (for example, to identify $2x + y$ with $x + y + 0 + x$), and the standard mechanisms for selecting the most promising clause and literal were tuned to our application [1, 2]. Early versions of MetiTarski performed well despite having only a modest amount of specialist code. By now, however, we have extended MetiTarski's code base extensively. We introduced case-splitting with backtracking [4], as is found in SMT solvers. We also included our own code for interval constraint solving, to either supplement or replace the external decision procedures.

MetiTarski relies on collections of upper and lower bounds (consisting of polynomials or rational functions) for the various special functions. The main effort in 2009 focused on refining these bounds, in particular through the introduction of continued fractions. Resolution chooses which axioms to use in a proof automatically. A single proof may use different axioms to cover different intervals of the region under consideration. A further benefit of our use of standard resolution is that other forms of axioms (concerning the absolute value function, or the min and max functions) can be written in first-order logic. The absolute value axioms state the obvious properties:

$$\neg(0 \leq x) \lor |x| = x \qquad 0 \leq x \lor |x| = -x$$

Resolution performs the appropriate sign reasoning automatically.

Resolution operates on *clauses*, or disjunctions of *literals*, which for MetiTarski are typically real inequalities. Under certain circumstances, MetiTarski can simplify the selected disjunction by formulating a problem that it can submit to an external decision procedure. Such problems involve a particular inequality in a disjunction, within its context. This context consists of the remainder of the disjunction and certain global facts. If the decision procedure finds the conjunction of these assertions to be inconsistent, then the inequality can be deleted from the clause. This connection between the basic resolution method and an external decision procedure is the key idea. Given another application domain, other decision procedures could probably be substituted for those called by MetiTarski. The only difficulty is that such a system would probably have to compete head-on with SMT solvers, which are highly refined and effective.

MetiTarski also uses the decision procedure for a form of redundancy elimination. As the proof search proceeds, polynomial formulas accumulate, and these are supplied to every decision procedure call. But if some of these formulas are redundant, they slow down subsequent calls without providing any benefit. Therefore, every time a new polynomial formula emerges from the resolution process, it is tested for redundancy—does it follow in the theory of RCF from previously known formulas?—and possibly discarded. The poor complexity of RCF quantifier elimination makes this step necessary.

3 MetiTarski's Decision Procedures

We adopted QEPCAD originally because it was free and easy to use, dedicated as it was to the single task of quantifier elimination. Moreover, QEPCAD worked extremely well in our first experiments. But QEPCAD had a number of limitations, concerning both portability (the code base seems to date from the distant past) and performance. Our decision problem is inherently intractable: doubly exponential in the number of variables in the problem [8]. This caused no difficulties at first, when virtually our entire problem set was univariate, but there are other ways to settle univariate special-function inequalities, and many important problems involve multiple variables.

Mathematica, the well-known computer algebra system, was next to be integrated with MetiTarski, as an alternative to QEPCAD. Though we regret the reliance on commercial software, many institutions already have Mathematica licences, and its quantifier elimination procedure is much more modern and powerful than QEPCAD's. It copes with problems in up to five variables, where QEPCAD cannot be expected to terminate at all. Mathematica has many configurable options, leaving us with many possible refinements to investigate. Mathematica can solve many special-function inequalities itself, and MetiTarski can take advantage of this capability to solve even harder problems.

The theorem prover Z3 [9], with its new extension for non-linear arithmetic [17], provides the third of our decision procedures. The great advantage for us is the possibility of working with its developers. We can tune it to our specific needs. Where it performs badly, we can send the problems for examination and know that they will be looked at. In some cases, Z3 has coped with problems in up to nine variables [21]. Z3 is free to non-commercial users.

Much of the effort needed to integrate different systems concerns overcoming conceptually trivial but serious obstacles. For many months, our team struggled with mysterious failures involving QEPCAD. These mainly happened during lengthy, overnight regression testing, where certain jobs would mysteriously hang and eventually bring all testing to a halt. Eventually, the problem was isolated to one of QEPCAD's peculiarities: unless it is used at a normal terminal, it performs its own echoing of input lines. (This allowed it to produce a readable output transcript when running in batch mode.) Because the inputs to QEPCAD can exceed 50K characters, and because MetiTarski never reads the output of QEPCAD until after it has sent a full problem to it, QEPCAD's output buffer would fill up, blocking its execution. Similar difficulties involving the other decision procedures take a surprising amount of time to diagnose and fix. Today as I write this, we are struggling with a mysterious problem plaguing integration with Z3.

Note that the choice among these three decision procedures is not straightforward. QEPCAD performs best in many situations.

4 Ongoing Research

We can often get better results if we do not regard the reasoning components of our system as black boxes. Automatically generated problems tend to be regular, and should if possible be tailored to the strengths of the component that will process them, or conversely, that component could itself be modified to perform better on those automatically generated problems. In the case of Z3, we were able to find a number of refinements that greatly improved its performance with MetiTarski [20]. One such refinement is to switch off a processing stage (univariate polynomial factorisation) that we could predict to be unnecessary. Another refinement, called *model sharing*, involved Z3 passing counterexamples to MetiTarski that it could use to eliminate some future Z3 calls.

Choosing which of the arithmetic solvers to call, given a particular problem, is itself a research question. A Cambridge student, Zongyan Huang, is currently investigating whether machine learning can be effective here. The basic idea is that features present in the special-function problem originally given to MetiTarski may be sufficient to predict which decision procedure will perform best on the polynomial decision problems that MetiTarski will generate for that problem. Features that we are examining include which special functions are present and how many variables there are. Zongyan is using Support Vector Machines (SVMs). This modelling approach, implemented as SVM-Light [16], is a form of machine learning that offers good results with reasonable efficiency. Her work is still experimental, but if it is successful, then realising it would involve MetiTarski running some machine learning code near the beginning of its execution.

MetiTarski opens the possibility of verifying dynamical systems using nonlinear models involving transcendental functions. Such models are common in engineering, for example in problems involving rotation. William Denman is investigating this area. He uses Mathematica (manually) to derive differential equations to model a given dynamical system. Such a model is a system of differential equations. Denman has written a Python program based on the algorithm implemented in HybridSAL [23], which is a tool for creating discrete models of hybrid systems. His program transforms the system of differential equations into a set of MetiTarski problems. MetiTarski is used to identify infeasible states in the abstract model, thereby simplifying it; the attraction of this approach is that it does not require MetiTarski to solve all the problems. The outcome of this procedure is a discrete, finite model suitable for model checking (currently, using NuSMV [6]).

5 Prospects for Further Integration

KeYmaera is a sophisticated interactive theorem prover designed for verifying hybrid systems [22]. We have recently joined MetiTarski to KeYmaera as a backend, hoping to provide the possibility of verifying systems whose models involve special functions. PVS is an interactive theorem prover designed for a variety of application areas, including hardware and hybrid systems [19]. William Denman, in collaboration with César Muñoz, has created an experimental linkup between MetiTarski and PVS. In both cases, the calling system invokes MetiTarski and trusts the result. These experiments should help identify new application areas for MetiTarski, suggesting areas for further development as well as providing justification for the effort needed to build a more robust integration. MetiTarski returns machine-readable proofs that combine standard resolution steps with a few additional inference rules, reflecting its use of computer algebra computations steps. These proofs can be used to facilitate the integration of MetiTarski with other systems, even if MetiTarski's conclusions are not trusted. In such applications, MetiTarski becomes a hub lying at the centre of a network of communicating reasoners.

The motivation for this research, years ago, was to equip Isabelle (an interactive theorem prover [18]) with support for reasoning about special functions. The original idea was to use lightweight methods that could prove relatively easy theorems. MetiTarski can prove difficult theorems, but through heavyweight methods that are difficult to include in an LCF-style theorem prover such as Isabelle. In such theorem provers, there is a strong preference to use only tools that justify every step in the proof kernel; so-called oracles that trust an external reasoner are frowned upon. The PVS community is more accommodating to oracles, and the present linkup between PVS and MetiTarski will be invaluable for investigating the potential of such combined systems. An integration with Sage [12], an open-source computer algebra system, is also planned for the near future.

Acknowledgements. Other members of the team include (in Edinburgh) Paul Jackson, Grant Passmore and Andrew Sogokon, and (in Cambridge) James Bridge, William Denman and Zongyan Huang. In the text above, "we" refers to all of us. We are grateful to our outside collaborators such as Eva Navarro, André Platzer, and many others not listed here.

The research was supported by the Engineering and Physical Sciences Research Council [grant numbers EP/C013409/1, EP/I011005/1, EP/I010335/1].

References

1. Behzad Akbarpour and Lawrence Paulson. MetiTarski: An automatic prover for the elementary functions. In Serge Autexier et al., editors, *Intelligent Computer Mathematics*, LNCS 5144, pages 217–231. Springer, 2008.
2. Behzad Akbarpour and Lawrence Paulson. MetiTarski: An automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 44(3):175–205, March 2010.
3. Michael Beeson. Automatic generation of a proof of the irrationality of e. *Journal of Symbolic Computation*, 32(4):333–349, 2001.
4. James Bridge and Lawrence Paulson. Case splitting in an automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 2012. In press; online at `http://dx.doi.org/10.1007/s10817-012-9245-6`.
5. Christopher W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *SIGSAM Bulletin*, 37(4):97–108, 2003.
6. A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV version 2: An opensource tool for symbolic model checking. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Computer Aided Verification*, LNCS 2404, pages 359–364. Springer, 2002.
7. Edmund Clarke and Xudong Zhao. Analytica: A theorem prover for Mathematica. *Mathematica Journal*, 3(1):56–71, 1993.
8. J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. Symbolic Comp.*, 5:29–35, 1988.

9. Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.
10. Andreas Dolzmann, Thomas Sturm, and Volker Weispfenning. Real quantifier elimination in practice. In B.Heinrich Matzat, Gert-Martin Greuel, and Gerhard Hiss, editors, *Algorithmic Algebra and Number Theory*, pages 221–247. Springer, 1999.
11. Lou van den Dries. Alfred Tarski's elimination theory for real closed fields. *Journal of Symbolic Logic*, 53(1):7–19, 1988.
12. M.A. Gray. Sage: A new mathematics software system. *Computing in Science Engineering*, 10(6):72–75, 2008.
13. Hoon Hong. QEPCAD — quantifier elimination by partial cylindrical algebraic decomposition. Sources and documentation are on the Internet at `http://www.cs.usna.edu/~qepcad/B/QEPCAD.html`.
14. Joe Hurd. First-order proof tactics in higher-order logic theorem provers. In Myla Archer, Ben Di Vito, and César Muñoz, editors, *Design and Application of Strategies/Tactics in Higher Order Logics*, number NASA/CP-2003-212448 in NASA Technical Reports, pages 56–68, September 2003.
15. Joe Hurd. Metis first order prover. Website at `http://gilith.com/software/metis/`, 2007.
16. Thorsten Joachims. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, pages 169–184. MIT Press, 1999.
17. Dejan Jovanovic and Leonardo Mendonça de Moura. Solving non-linear arithmetic. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *IJCAR 2012*, LNCS 7364, pages 339–354. Springer, 2012.
18. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002. LNCS Tutorial 2283.
19. S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. PVS: Combining specification, proof checking, and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification: 8th International Conference, CAV '96*, LNCS 1102, pages 411–414. Springer, 1996.
20. Grant Passmore, Lawrence Paulson, and Leonardo de Moura. Real algebraic strategies for MetiTarski proofs. In Johan Jeuring, John Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics*, volume 7362 of *Lecture Notes in Computer Science*, pages 358–370. Springer, 2012.
21. Lawrence C. Paulson. MetiTarski: Past and future. In Lennart Beringer and Amy P. Felty, editors, *ITP*, LNCS 7406, pages 1–10. Springer, 2012.
22. André Platzer and Jan-David Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning — 4th International Joint Conference, IJCAR 2008*, LNCS 5195, pages 171–178. Springer, 2008.
23. Ashish Tiwari. HybridSAL relational abstracter. In P. Madhusudan and Sanjit A. Seshia, editors, *Computer Aided Verification*, volume 7358 of *Lecture Notes in Computer Science*, pages 725–731. Springer Berlin Heidelberg, 2012.