MetiTarski: Past and Future

Lawrence C. Paulson

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

Abstract. A brief overview is presented of MetiTarski [4], an automatic theorem prover for real-valued special functions: ln, exp, sin, cos, etc. MetiTarski operates through a unique interaction between decision procedures and resolution theorem proving. Its history is briefly outlined, along with current projects. A simple collision avoidance example is presented.

1 Introduction

MetiTarski [4] is an automatic theorem prover for first-order logic over the real numbers, including the transcendental and other special functions. It is a version of Joe Hurd's Metis [23, 24] (a resolution theorem prover) heavily modified to call decision procedures, and more recently, augmented with case-splitting by backtracking.

Here are some theorems that MetiTarski can prove, automatically of course, and typically in tens of seconds.

$$\begin{aligned}&\forall\, t > 0, v > 0\\ &\quad ((1.565 + 0.313\, v)\, \cos(1.16\, t) + (.0134 + .00268\, v)\, \sin(1.16\, t))\, e^{-1.34\, t}\\ &\qquad\qquad - (6.55 + 1.31\, v)\, e^{-0.318\, t} + v \;\geq\; -10\end{aligned}$$

$$\forall\, x > 0 \Longrightarrow \frac{1 - e^{-2\,x}}{2\,x\,(1 - e^{-x})^2} - \frac{1}{x^2} \leq \frac{1}{12}$$

$$\forall\, x\, y,\ x \in (0, 12) \Longrightarrow xy \leq \frac{1}{5} + x\, \ln(x) + e^{y-1}$$

$$\forall\, x \in (-8, 5) \Longrightarrow \max(\sin(x), \sin(x + 4), \cos(x)) > 0$$

$$\forall x\, y,\ (0 < x < y \wedge y^2 < 6) \Longrightarrow \frac{\sin(y)}{\sin(x)} \leq 10^{-4} + \frac{y - \frac{1}{6}y^3 + \frac{1}{120}y^5}{x - \frac{1}{6}x^3 + \frac{1}{120}x^5}$$

$$\forall x \in (0, 1) \Longrightarrow 1.914\frac{\sqrt{1+x} - \sqrt{1-x}}{4 + \sqrt{1+x} + \sqrt{1-x}} \leq 0.01 + \frac{x}{2 + \sqrt{1 - x^2}}$$

$$\forall x \in (0, 1.25) \Longrightarrow \tan(x)^2 \leq 1.75\, 10^{-7} + \tan(1)\tan(x^2)$$

$$\forall x \in (-1, 1), y \in (-1, 1) \Longrightarrow \cos(x)^2 - \cos(y)^2 \leq -\sin(x + y)\sin(x - y) + 0.25$$

$$\forall x \in (-1,1), y \in (-1,1) \Longrightarrow \cos(x)^2 - \cos(y)^2 \geq -\sin(x+y)\sin(x-y) - 0.25$$

$$\forall x \in (-\pi,\pi) \Longrightarrow 2\,|\sin(x)| + |\sin(2\,x)| \leq \frac{9}{\pi}$$

The motivation for MetiTarski arose with Jeremy Avigad's Isabelle/HOL proof of the prime number theorem. Avigad [7, Sect. 4.5] observed that some quite elementary inequalities involving the logarithm function (the proof requires establishing many such inequalities) were inordinately difficult to prove. My original idea was to create a simple-minded heuristic procedure within Isabelle to prove inequalities involving continuous functions. It might reason using monotonicity, backwards chaining, and ideas similar to those in the Fourier-Motzkin [26] approach to deciding real inequalities. Although decision procedures are popular, reasoning about arbitrary special functions is obviously undecidable. The decidable problems that do exist are very complex, both in theory and in practice. Our approach has to be heuristic.

2 Early Work

The initial work was done (using funding from the UK's EPSRC) by my colleague, Behzad Akbarpour. He located a promising paper [29] (see also Daumas et al. [16]) giving formally verified upper and lower bounds (polynomials or rational functions, namely, ratios of polynomials) for the well-known transcendental functions and describing techniques using interval arithmetic to establish ground inequalities involving special functions over the real numbers. Hand simulations quickly established that interval arithmetic was seldom effective at solving problems even in one variable [5].

Equipped with the upper and lower bounds, we could replace the transcendental functions appearing in a problem by polynomials. We could therefore reduce the original special-function inequality to a set of polynomial inequalities. Because interval arithmetic was too weak to decide these inequalities, we decided to try a decision procedure. First-order formulas over polynomial inequalities over the real numbers admit quantifier elimination [20], and are therefore decidable. This decision problem is known as RCF, for real closed fields.

The first implementation of MetiTarski [2] used John Harrison's implementation [27] of the Cohen-Hörmander RCF decision procedure. It turned out to be much more effective than interval arithmetic. As the procedure was itself coded in ML, MetiTarski was a self-contained ML program. Unfortunately, we found [2] that the Cohen-Hörmander procedure couldn't cope with polynomials of degree larger than five, which ruled out the use of accurate bounds. The next version of MetiTarski [3] invoked an external decision procedure, QEPCAD [12], which implements a much more powerful technique: cylindrical algebraic decomposition (CAD). More recently, we have integrated MetiTarski with the computer algebra system Mathematica, which contains a family of highly advanced RCF decision procedures. We can even use the SMT solver Z3 [18] now that it has been extended with nlsat, a novel approach to deciding purely existential RCF problems [25].

A separate question concerned how to apply the upper and lower bounds to eliminate special function occurrences. Should we write a bespoke theorem prover implementing a carefully designed algorithm? Analytica [14] and Weierstrass [8] both implement a form of sequent calculus. Despite the impressive results obtained by both of these systems, we decided to see whether ordinary first-order resolution could perform as well. The contest between "natural" and resolution proof systems is a long-standing scientific issue in automated reasoning [10]. Resolution had two strong arguments in its favour: one, its great superiority (in terms of performance) over any naive implementation of first-order logic, and two, the possibility that resolution might also be superior to a bespoke algorithm at finding complicated chains of reasoning involving the upper and lower bounds.

Resolution has entirely met our expectations. No individual bound can accurately approximate a special function over the entire real line. This was already clear in Muñoz and Lester [29], who presented families of bounds, each accurate over a small interval. A typical proof involves choosing multiple bounds over overlapping intervals of the real line. With resolution, we can supply bounds as files of axioms. Resolution automatically identifies bounds that are accurate enough, keeping track of the intervals for which the theorem has been proved. Moreover, other facts (such as the definitions of the functions abs and max) can be asserted declaratively as axioms.

The original families of bounds also underwent many refinements. Through careful scrutiny, we were able to extend their ranges of applicability. A later version of MetiTarski [4] adopted many continued fraction approximations [15]. For some functions, notably ln and $\tan^{-1}$, continued fractions are vastly more accurate than truncated Taylor series. There is also the choice between inaccurate but simple bounds (linear ones are helpful if functions are nested), and highly accurate bounds of high degree. Although the most recent version of MetiTarski can choose axiom files automatically, manually choosing which bounds to include can make a difference between success and failure for some problems. To extend MetiTarski to handle a new function, the most important step is to find approximations to the function that yield suitable upper and lower bounds.

3 Basic Architecture

MetiTarski comprises a modified resolution theorem prover, one or more RCF decision procedures and a collection of axiom files giving approximations (upper and lower bounds) to special functions. A few other axioms are used, relating division with multiplication, defining the absolute value function, etc.

The most important modifications to the resolution method are arithmetic simplification and the interface to RCF decision procedures. Polynomials are simplified in an ad hoc manner, loosely based on Horner canonical form; this serves first to identify obviously equivalent polynomials, but second and crucially to identify a special function occurrence to be eliminated. The mathematical formula is transformed to move this candidate occurrence into a certain position, which will allow an ordinary resolution step to replace it by an upper or lower bound, as appropriate. Occurrences of the division operator are also simplified and in particular flattened, so that an algebraic expression contains no more than a single occurrence of division, and that outermost.

RCF decision procedures are mainly used to simplify clauses by deleting certain literals. Recall that a clause is a disjunction of literals, each of which is an atomic formula or its negation. With standard resolution, a literal can only be deleted from a clause after a resolution step with another clause containing that literal's negation. MetiTarski provides another way a literal can be deleted: whenever an RCF decision procedure finds it to be inconsistent. This determination is done with respect to the literal's context, consisting of the negation of other literals in the same clause, as well as any globally known algebraic facts.

RCF decision procedures are also used to discard redundant clauses. Whenever a new clause emerges from the resolution process, it is given to the decision procedure, and if the disjunction of its algebraic literals turns out to be a logical consequence (in RCF) of known facts, then this clause is simply ignored. This step prevents the buildup of redundant algebraic facts, which cannot influence future decision procedure calls.

Our use of RCF decision procedures has one massive disadvantage: performance. The general decision problem is doubly exponential in the number of variables [17]. We actually use a special case of the decision problem, in which we ask whether universally quantified formulas are theorems, but even so, when a proof takes a long time, almost invariably this time is spent in decision procedures. The time spent in resolution is frequently just a few seconds, a tiny fraction of the total.

We undertook this project as basic research, motivated by Avigad's difficulties but with no other specific applications. Having no problem set to begin with, we (mainly Akbarpour) created our own. The original problems mostly came from mathematical reference works [1, 13, 28]. Later, we formalised engineering problems, including Nichols plot problems from Ruth Hardy's thesis [21], and simple hybrid systems problems that were published on a website [35]. The hardware verification group at Concordia University supplied a few more problems. We now have nearly 900.

4 Current Projects

Recent developments fall under three categories: improvements to the resolution process, the use of new decision procedures, and applications.

Resolution can be tweaked in countless ways, but one of the most important heuristics is splitting. This involves taking a clause of the form $A \lor B$ (all nontrivial clauses are disjunctions) and considering the cases A and B as separate problems. Given that resolution already can deal with disjunctions, splitting is not always appropriate, and indeed splitting is only possible if A and B have no variables in common. We have experimented [11] with two kinds of splitting: lightweight (essentially a simulation, implemented by adding propositional variables as labels in clauses) and with backtracking (similar to the splitting done by SAT-solvers). Backtracking is complicated to implement in the context of resolution theorem proving: it affects many core data structures, which must now be saved and restored according to complicated criteria; this work was done by James Bridge. Both forms of splitting deliver big performance improvements. The problem is only split into separate cases when each involves a special function problem.

The decision procedure is a crucial element of MetiTarski. From the first experiments using interval arithmetic and the first implementation, using the Cohen-Hörmander method, each improvement to the decision procedure has yielded dramatic improvements overall. We have the most experience with QEPCAD, but in the past year we have also used Mathematica and Z3. Each has advantages. QEPCAD is open source and delivers fast performance on problems involving one or two variables. Unfortunately, QEPCAD is almost unusable for problems in more than three variables, where Mathematica and Z3 excel. Grant Passmore is heavily involved with this work. With Mathematica, we can solve problems in five variables, and with Z3, we have sometimes gone as high as nine. (See the example in Sect. 6.) But we do not have the luxury of regarding the decision procedure as a black box. That is only possible when the performance bottlenecks are elsewhere. We have to examine the sort of problems that MetiTarski produces for the decision procedures, and configure them accordingly. Such tuning delivers significant improvements [30].

5 Applications

We have assumed that, as MetiTarski's performance and scope improved, users would find their own applications. Meanwhile, we have ourselves investigated applications connected with hybrid systems, and in particular with stability properties. Note that MetiTarski is not itself a hybrid system verifier: it knows nothing about the discrete states, transitions and other elements of a hybrid system. The discrete aspect of a hybrid system must be analysed by other means, but MetiTarski can assist in verifying the continuous aspect [6]. There have also been a few simple experiments involving the verification of analogue circuits [19].

The ability to prove theorems combining first-order logic and real-valued special functions is unique to MetiTarski. Mathematica can establish such properties in simple cases, as can certain constraint solvers [34], but these tools do not deliver proofs. MetiTarski delivers resolution proofs, which specify every detail of the reasoning apart from the arithmetic simplification and the decision procedure calls. Even these reasoning steps do not necessarily have to be trusted, as discussed below (Sect. 7).

Mathematicians will find that MetiTarski proofs are seldom natural or elegant. Mathematical properties of a special function are typically proved from first principles, referring to the function's definition and appealing to general theorems. A MetiTarski proof is typically a complicated case analysis involving various bounds of functions and signs of divisors. Compared with a mathematician's proof, such a proof will frequently yield a less general result (perhaps limited to a narrow interval). For these reasons, MetiTarski is more appropriate for establishing inequalities that arise in engineering applications, inequalities that hold for no simple reason.

6 A Collision Avoidance Problem

A problem studied by Platzer [32, Sect. 3.4] concerns collision avoidance for two aircraft, named x and y. For simplicity, we consider only two dimensions: the aircraft are flying in the XY plane.

The coordinates of aircraft x are written (x_1, x_2). Here, the subscript 1 refers to the X component of the aircraft's position, while subscript 2 refers to the Y component. (I am not to blame for this confusing notation!) Similarly, the velocity vector in two dimensions is written (d_1, d_2).

Glossing over the details of the derivation, here is a summary of the system of equations governing this aircraft:

$$\begin{array}{llll} x_1'(t) = d_1(t) & x_2'(t) = d_2(t) & d_1'(t) = -\omega d_2(t) & d_2'(t) = \omega d_1(t) \\ x_1(0) = x_{1,0} & x_2(0) = x_{2,0} & d_1(0) = d_{1,0} & d_2(0) = d_{2,0} \end{array}$$

This system admits a closed-form solution, yielding the trajectory of aircraft x:

$$\begin{aligned} x_1(t) &= x_{1,0} + \frac{d_{2,0}\cos(\omega t) + d_{1,0}\sin(\omega t) - d_{2,0}}{\omega} \\ x_2(t) &= x_{2,0} - \frac{d_{1,0}\cos(\omega t) - d_{2,0}\sin(\omega t) - d_{1,0}}{\omega} \end{aligned}$$

The treatment of aircraft y, whose coordinates are written (y_1, y_2), is analogous. We would like to prove that two aircraft following the trajectory equations, for certain ranges of initial locations and linear velocities will maintain a safe distance (called p, for "protected zone"):

$$(x_1(t) - y_1(t))^2 + (x_2(t) - y_2(t))^2 > p^2$$

Figure 1 presents the corresponding MetiTarski input file. MetiTarski can prove the theorem, but the processor time is 924 seconds.[1] Of this, six seconds are devoted to resolution proof search and the rest of the time is spent in the RCF decision procedure (in this case, Z3). The problem is difficult for MetiTarski because of its nine variables. It is only feasible because of our recent research [30] into heuristics such as model sharing (which can eliminate expensive RCF calls by utilising information obtained from past calls) and specific strategies that fine-tune Z3 to the problems that MetiTarski gives it.

Naturally, users would like to handle problems in many more variables. This is the biggest hurdle in any application of RCF decision procedures.

[1] On a Mac Pro Dual Quad-core Intel Xeon, 2.8 GHz, with 10GB RAM

```
fof(airplane_easy,conjecture,
  (! [T,X10,X20,Y10,Y20,D10,D20,E10,E20] :
    (
      (
        0 < T & T < 10 & X10 < -9 & X20 < -1 & Y10 > 10 & Y20 > 10 &
        0.1 < D10 & D10 < 0.15 & 0.1 < D20 & D20 < 0.15 &
        0.1 < E10 & E10 < 0.15 & 0.1 < E20 & E20 < 0.15
      )
      =>
      (
        (X10 - Y10 - 100*D20 - 100*E20 + (100*D20 + 100*E20)*cos(0.01*T)
        + (100*D10 - 100*E10)*sin(0.01*T))^2 +
        (X20 - Y20 + 100*D10 + 100*E10 + (-100*D10 - 100*E10)*cos(0.01*T)
        + (100*D20 - 100*E20)*sin(0.01*T))^2
      )
      > 2
    )
  )
).
include('Axioms/general.ax').
include('Axioms/sin.ax').
include('Axioms/cos.ax').
```

Fig. 1. Aircraft Collision Avoidance

Platzer's treatment of this problem is quite different. He uses KeyMaera, his hybrid system verifier [33]. KeyMaera models a hybrid system that controls the aircraft, while we examine only a part of the continuous dynamics of this system. Even the continuous dynamics are treated differently: KeyMaera has a principle called *differential induction* [31] that can establish properties of the solutions to differential equations without solving them. To prepare these MetiTarski problems, the differential equations must first be solved (perhaps using Mathematica), yielding explicit formulas for the aircrafts' trajectories. KeyMaera can verify much larger hybrid systems than MetiTarski can. However, recall that MetiTarski is not specifically designed for verifying this type of example: it is simply a general-purpose theorem prover for the reals.

7 Integration with Proof Assistants

As mentioned before, MetiTarski combines classical resolution with arithmetic simplification, RCF decision procedures and axioms describing the behaviour of various real-valued functions. It returns machine-readable proofs that combine standard resolution steps with these extensions.

If MetiTarski is to be added to an interactive theorem prover as a trusted oracle, little effort is required other than to ensure that all problems submitted to it are first-order with all variables ranging over the real numbers. To reduce the level of trust required, MetiTarski proofs could be broken down into their various elements and reconstructed in Isabelle, leaving only the most intractable steps to oracles.

– Arithmetic simplification in MetiTarski involves little more than reducing polynomials to canonical form and extending the scope of the division operator; these steps should be easy to verify by automation in an LCF-style interactive theorem prover.
– The axioms used by MetiTarski are simply mathematical facts that could, in principle, be developed in any capable interactive theorem prover. Formal developments of the familiar power series expansions already available [16]. However, the theory of continued fractions seems to rest on substantial bodies of mathematics that are yet to be mechanised. Possibly some of these bounds could be verified individually, independently of the general theory.
– The most difficult obstacle is that of the RCF decision procedure. Unfortunately, the algorithms are complicated and we have few implementations to choose from. Until now there has been little interest in procedures that justify their answers. However, Harrison has investigated sum of squares techniques that could produce certificates for such proofs eventually [22].

One fact in our favour is that MetiTarski's entire proof search does not have to be justified, only the final proof, which represents a tiny fraction of the reasoning.

Before undertaking such an integration, it is natural to ask how many potential applications there are. It is sobering to consider that after SMT solvers were integrated with Isabelle, they were left essentially unused [9], and SMT solvers are much more generally applicable than MetiTarski. This integration might be included in a larger project to verify a specific and substantial corpus of continuous mathematics.

8 Conclusion

Research on MetiTarski is proceeding in many directions. Improvements to the use of decision procedures are greatly increasing MetiTarski's scope and power, especially by increasing the number of variables allowed in a problem. Meanwhile, many new kinds of applications are being examined. Integrating MetiTarski with an interactive theorem prover such as Isabelle is not straightforward, but is feasible given motivation and resources.

Acknowledgements. Eva Navarro has made a great effort to give us problems in control engineering to examine. Other members of the team include Paul Jackson and Andrew Sogokon (in Edinburgh), James Bridge, Grant Passmore, William Denman and Zongyan Huang. Denman contributed the collision avoidance example. Bridge and Passmore commented on drafts of this paper, as did Lennart Beringer.

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. Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Wiley, 1972.
2. Behzad Akbarpour and Lawrence Paulson. Extending a resolution prover for inequalities on elementary functions. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 47–61, 2007.
3. 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.
4. 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.
5. Behzad Akbarpour and Lawrence C. Paulson. Towards automatic proofs of inequalities involving elementary functions. In Byron Cook and Roberto Sebastiani, editors, *PDPAR: Pragmatics of Decision Procedures in Automated Reasoning*, pages 27–37, 2006.
6. Behzad Akbarpour and Lawrence C. Paulson. Applications of MetiTarski in the verification of control and hybrid systems. In Rupak Majumdar and Paulo Tabuada, editors, *Hybrid Systems: Computation and Control*, LNCS 5469, pages 1–15. Springer, 2009.
7. Jeremy Avigad, Kevin Donnelly, David Gray, and Paul Raff. A formally verified proof of the prime number theorem. *ACM Transactions on Computational Logic*, 9(1), 2007.
8. Michael Beeson. Automatic generation of a proof of the irrationality of e. *JSC*, 32(4):333–349, 2001.
9. Jasmin Christian Blanchette, Sascha Böhme, and Lawrence C. Paulson. Extending Sledgehammer with SMT solvers. In Nikolaj Børner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction — CADE-23*, volume 6803 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2011.
10. W. W. Bledsoe. Non-resolution theorem proving. *Artificial Intelligence*, 9:1–35, 1977.
11. 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.
12. Christopher W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *SIGSAM Bulletin*, 37(4):97–108, 2003.
13. Peter Southcott Bullen. *A Dictionary of Inequalities.* Longman, 1998.
14. Edmund Clarke and Xudong Zhao. Analytica: A theorem prover for Mathematica. *Mathematica Journal*, 3(1):56–71, 1993.
15. A. Cuyt, V. Petersen, B. Verdonk, H. Waadeland, and W.B. Jones. *Handbook of Continued Fractions for Special Functions.* Springer, 2008.
16. Marc Daumas, César Muñoz, and David Lester. Verified real number calculations: A library for integer arithmetic. *IEEE Trans. Computers*, 58(2):226–237, 2009.
17. J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. Symbolic Comp.*, 5:29–35, 1988.
18. 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.

19. William Denman, Behzad Akbarpour, Sofiène Tahar, Mohamed Zaki, and Lawrence C. Paulson. Formal verification of analog designs using MetiTarski. In Armin Biere and Carl Pixley, editors, *Formal Methods in Computer Aided Design*, pages 93–100. IEEE, 2009.
20. Lou van den Dries. Alfred tarski's elimination theory for real closed fields. *The Journal of Symbolic Logic*, 53(1):7–19, 1988.
21. Ruth Hardy. *Formal Methods for Control Engineering: A Validated Decision Procedure for Nichols Plot Analysis.* PhD thesis, University of St Andrews, 2006.
22. John Harrison. Verifying nonlinear real formulas via sums of squares. In Klaus Schneider and Jens Brandt, editors, *Theorem Proving in Higher Order Logics: TPHOLs 2007*, LNCS 4732, pages 102–118. Springer, 2007.
23. 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.
24. Joe Hurd. Metis first order prover. Website at http://gilith.com/software/metis/, 2007.
25. Dejan Jovanoviċ and Leonardo de Moura. Solving Non-linear Arithmetic. Technical Report MSR-TR-2012-20, Microsoft Research, 2012. Accepted to IJCAR'12.
26. J.-L. Lassez and M. J. Maher. On fourier's algorithm for linear arithmetic constraints. *Journal of Automated Reasoning*, 9(3):373–379, 1992.
27. Sean McLaughlin and John Harrison. A proof-producing decision procedure for real arithmetic. In Robert Nieuwenhuis, editor, *Automated Deduction — CADE-20 International Conference*, LNAI 3632, pages 295–314. Springer, 2005.
28. Dragoslav S. Mitrinović and Petar M. Vasić. *Analytic Inequalities.* Springer, 1970.
29. César Muñoz and David Lester. Real number calculations and theorem proving. In Joe Hurd and Tom Melham, editors, *Theorem Proving in Higher Order Logics: TPHOLs 2005*, LNCS 3603, pages 195–210. Springer, 2005.
30. Grant Olney Passmore, Lawrence C. Paulson, and Leonardo de Moura. Real algebraic strategies for MetiTarski proofs. In Johan Jeuring, editor, *Conferences on Intelligent Computer Mathematics — CICM 2012.* Springer, 2012. In press.
31. André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Logic Computation*, 20(1):309–352, 2010.
32. André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics.* Springer, 2010.
33. 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.
34. Stefan Ratschan. Efficient solving of quantified inequality constraints over the real numbers. *ACM Trans. Comput. Logic*, 7(4):723–748, 2006.
35. Stefan Ratschan and Zhikun She. Benchmarks for safety verification of hybrid systems, 2008. http://hsolver.sourceforge.net/benchmarks/.