Automated Real Proving in PVS via MetiTarski

William Denman[1]* and César Muñoz[2]

[1] University of Cambridge, Computer Laboratory, UK, wd239@cam.ac.uk
[2] NASA, Langley Research Center, US, cesar.a.munoz@nasa.gov

Abstract. This paper reports the development of a proof strategy that integrates the MetiTarski theorem prover as a trusted external decision procedure into the PVS theorem prover. The strategy automatically discharges PVS sequents containing real-valued formulas, including transcendental and special functions, by translating the sequents into first order formulas and submitting them to MetiTarski. The new strategy is considerably faster and more powerful than other strategies for non-linear arithmetic available to PVS.

1 Introduction

Formally reasoning about the behavior of safety-critical cyber-physical systems is a difficult and well-known problem. To address the verification of these real-world systems, state-of-the-art formal tools should be able to reason about more than just polynomial functions. MetiTarski [1] is an automated theorem prover for first order formulas containing inequalities between transcendental and special functions such as sin, cos, exp, sqrt, etc. A modified resolution framework guides the proof search, replacing instances of special functions by verified upper and lower polynomial bounds. During resolution, decision procedures for the theory of real closed fields (RCF) are called to delete algebraic clauses that are inconsistent with other derived facts. The current implementation of MetiTarski takes advantage of the highly-efficient non-linear satisfiability methods within the SMT solver Z3 for RCF decisions.

The Prototype Verification System (PVS) [8] is a formal verification environment that consists of a specification language, based on a classical higher-order logic enriched with an expressive type system, and an interactive theorem prover for this logic. The PVS specification language is strongly typed and supports predicate subtyping. In particular, the numerical types are defined such that `nat` (natural numbers) is a subtype of `int` (integers), `int` is a subtype of `rat` (rationals), `rat` is a subtype of `real` (reals), and `real` is a subtype of the primitive type `number`. The subtyping hierarchy of numerical types and the fact that

* Research supported by SRI International, under NSF Grant CNS-0917375, and Engineering and Physical Sciences Research Council, under grants EP/I011005/1 and EP/I010335/1. Author would like to thank the National Institute of Aerospace for a short visit supported by the Assurance of Flight Critical System's project of NASA's Aviation Safety Program at Langley Research Center under Research Cooperative Agreement No. NNL09AA00A.

rational arithmetic is built-in makes PVS well suited for real number proving. In particular, ground numerical expressions are automatically (and efficiently) simplified by the PVS theorem prover. For example, the numerical expression `1/3+1/3+1/3` is simplified to `1` and this simplification *does not* require a proof. PVS has been extensively used at NASA in the formal verification of algorithms and operational concepts for the next generation of air traffic management systems.[3]

The NASA PVS Library[4], which is the de facto PVS standard library, includes several strategies for manipulating [3] and simplifying [5] real number formulas. The most advanced proof strategies for real number proving available in the NASA PVS Library are `interval` [2, 7] and `bernstein` [6]. These strategies are based on provably correct interval arithmetic and Bernstein polynomial approximations, respectively. The strategy `interval` automatically discharges sequent formulas involving transcendental and other special functions. The strategy `bernstein` automatically discharges simply-quantified multivariate polynomial inequalities. The main characteristic of these strategies is that they preserve soundness, i.e., proofs that use `interval` and `bernstein` can be expanded into a tree of primitive PVS proof rules. Unfortunately, this also means that these strategies are not as efficient as specialized theorems provers like MetiTarski.

For interactive theorem provers such as PVS, access to external decision procedures for the theory of real closed fields can greatly speed up the verification time of large and complex algorithms. This paper describes the integration of MetiTarski as a trusted oracle within PVS. This integration greatly improves the automated capabilities of PVS for proving properties involving real numbers.

2 The PVS Strategy `metit`

The proof strategy that integrates the RCF automated theorem prover MetiTarski into the PVS theorem prover is called `metit`. This strategy, which is currently available as part of the NASA PVS Library for PVS 6.0, requires MetiTarski and an external arithmetic decision procedure such as Z3.[5]

In its simplest form, the strategy `metit` can be used to prove universally-quantified formulas involving real numbers such as

$$\forall v \in [200, 250], |\phi| \leq 35 : \left| \frac{180\, g}{\pi v\, 0.514} \tan(\frac{\pi \phi}{180}) \right| < 3.825, \tag{1}$$

where $g = 9.8$ (gravitational acceleration in meters per second squared) and π is the well-known irrational constant. This formula, which appears in the formal verification of an alerting algorithm for parallel landing [4], states that for an aircraft flying at a ground speed between 200 and 250 knots and maximum bank angle of 35 degrees, the angular speed is less than 3.825 degrees per second.

[3] http://shemesh.larc.nasa.gov/fm/fm-atm-cdr.html.

[4] http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library.

[5] The full distribution of the NASA PVS Library includes pre-installed binaries of MetiTarski 2.2 and Z3 4.3.1 for Mac OSX 10.7.3 and 64-bits Linux.

Figure 1 shows Formula 1 as a sequent in PVS. The double hash symbol "`##`" is the inclusion operator of closed intervals, which are denoted using the parenthesis operator "`[| |]`". The sequent, which consists of one universally-quantified formula in the consequent, is automatically discharged by the proof strategy `metit` in less than one second. The strategy uses PVS' internal utilities to parse the sequent. If the sequent is recognized as a set of first order formulas involving real numbers, the strategy translates the sequent into a TPTP[6] formula and submits it to MetiTarski. If MetiTarski returns *SZS status Theorem*, the result is trusted by PVS and the sequent is closed. If MetiTarski returns *SZS status Timeout* or *SZS status GaveUp* then the sequent in question is returned back to PVS unchanged. Application of other proof strategies would be required at this stage.

```
  |-------
{1}   FORALL (v, phi:real): abs(phi) <= 35 AND v ## [|200, 250|] IMPLIES
        abs(180*9.8*tan(phi*pi/180)/(pi*v*0.514)) < 3.825

Rule? (metit)
Metitarski Input =
fof(pvs2metit,conjecture, (![V1, PHI2]: (((abs(PHI2) <= 35) & (200 <=
V1 & V1 <= 250)) => (abs((((180*(98/10))*tan(((PHI2*pi)/180)))/((pi*V1)
*(514/1000)))) < (3825/1000))))).
SZS status Theorem for tr_35.tptp
Processor time: 0.680 = 0.184 (Metis) + 0.496 (RCF)
Trusted source: MetiTarski.
Q.E.D.
```

Fig. 1. Automated proof of Formula 1 using `metit`

Although universally-quantified real-number formulas such as Formula 1 occur in the verification of complex systems, a more common use case for the strategy `metit` is in the context of an interactive proof of a large theorem where multiple formulas appear in a sequent. The strategy `metit` only deals with sequents that are sets of first order formulas containing real-number inequalities between transcendental and special functions. However, the user may optionally specify formulas of interest in a given sequent. Other formulas in the sequent will be ignored by the strategy.

Moreover, in an interactive theorem prover such as PVS, sequent formulas may also involve data structures such as records, arrays, tuples, and abstract data types. For example, the sequent in Figure 2 appears in a lemma that characterizes

[6] The TPTP format is used by the Thousands of Problems for Theorem Provers library (http://www.cs.miami.edu/~tptp.)

aircraft trajectories that are repulsive.[7] This sequent consists of 12 antecedent formulas and one consequent formula. All of the formulas are quantifier-free, but free-variables (Skolem constants, in PVS terminology) occurring in the sequent can be understood as universally-quantified variables. In addition to the real variable `eps`, this sequent involves record variables `v`, `rd`, `dv`, and `mps`, which represent vectors in a 2-D Euclidean space.

The strategy `metit` does not directly deal with data structures. However, it recognizes that an expression such as `v‘x`, which accesses the field `x` of 2-D vector variable `v`, denotes a real-number variable. Hence, the strategy appropriately translates record and tuple access expressions as variables in the TPTP syntax. Furthermore, the strategy `metit` allows the user to specify the formulas of interest that are to be sent to MetiTarski. The proof command `(metit *)`, where the asterisk symbol "`*`" specifies all formulas in the sequent, translates the 13 formulas of the sequent into a TPTP formula involving 9 variables. This particular TPTP formula is discharged by MetiTarski in less than 0.2 seconds.

Further analysis of the sequent in Figure 2 reveals that all the formulas in the sequent are necessary to discharge it. For example, the proof command `(metit (^ -1))`, where `(^ -1)` denotes all formulas in the sequent but the first one in the antecedent, does not succeed to prove the sequent. In total, the proof of the lemma where this particular sequent appears requires 171 invocations of `metit` and, including all the other proof rules, the lemma is proved in 37 seconds. The largest sequent discharged by `metit` in this proof involves 13 variables. It is important to note that none of these sequents can be discharged by any other automated strategies available to PVS.

The use case for the PVS strategy `metit` is ideally for lemmas containing transcendental functions and special functions. However proofs of purely polynomial problems can also take advantage of the integration of PVS, MetiTarski, and Z3. What distinguishes Z3 from other state-of-the-art SMT solvers is that its proof heuristics for non-linear real arithmetic are customizable through a strategy language. MetiTarski itself also implements its own set of strategies that work in combination with those of Z3.

3 Results and Conclusion

To test the capabilities of the integration of MetiTarski with PVS, the proof strategy `metit` was run on the suite of examples from the PVS contribution `interval_arith`.[8] These examples involve trigonometric and other special functions, which are not supported by the strategy `bernstein`. The experiments were run on an Intel Core2Duo 2.4GHz processor with 4GB of RAM. The results are

[7] Lemma `repulsive_criteria_iterative_reduces_seq_divergent_special` of theory `repulsive_iterative` in the contribution `ACCoRD` of the NASA PVS Library. Thanks to Anthony Narkawicz, NASA Langley, for providing this example.

[8] The test suite is available in the theory `metit_examples` in the contribution `MetiTarski` of the NASA PVS Library.

```
repulsive_criteria_iterative_reduces_seq_divergent_special.3.1.1.1 :
[-1]  eps = 1 OR eps = -1
[-2]  v`y*eps <= 0
[-3]  rd`y*eps < 0
[-4]  ((v`x = 0 AND v`y = 0) IMPLIES rd`x >= 0)
[-5]  ((v`x /= 0 OR v`y /= 0) IMPLIES rd`x > v`x)
[-6]  rd`x*v`y*eps-rd`y*v`x*eps <= 0
[-7]  mps`y*eps+rd`y*eps < 0
[-8]  v`x >= 0
[-9]  (dv`x /= 0 OR dv`y /= 0)
[-10] mps`x*rd`y*eps-mps`y*rd`x*eps <= 0
[-11] -1*(dv`x*mps`y*eps)-dv`x*rd`y*eps+ dv`y*mps`x*eps+dv`y*rd`x*eps < 0
[-12] ((rd`x*mps`x+rd`x*rd`x+rd`y*mps`y+rd`y*rd`y < 0 AND
      dv`x*rd`y*eps-dv`y*rd`x*eps < 0) OR (rd`x*mps`x+rd`x*rd`x+
      rd`y*mps`y+rd`y*rd`y >= 0 AND dv`x*mps`x+dv`x*rd`x+dv`y*mps`y+
      dv`y*rd`y > rd`x*mps`x+rd`x*rd`x+rd`y*mps`y+rd`y*rd`y
      AND dv`x*rd`y*eps-dv`y*rd`x*eps <= 0))
  |-------
[1] (dv`x /= 0 OR dv`y /= 0) AND dv`y*eps < 0 AND ((v`x = 0 AND v`y = 0)
    IMPLIES dv`x >= 0) AND ((v`x /= 0 OR v`y /= 0) IMPLIES dv`x > v`x)
    AND dv`x*v`y*eps-dv`y*v`x*eps <= 0
```

Fig. 2. Sequent involving 13 formulas and 9 variables

displayed in Table 1. Each row is a separate attempt to prove the specified lemmas. The next two columns each list the total proof time for the respective proof strategy. On average, the speed up to proof times was on the factor of 18. In an interactive proof where multiple sub-problems of the type listed in Table 1 occur, the potential reduction in overall proof time is substantial. However, it should be noted that while `interval` is a proof-producing strategy, i.e., `interval` preserves the soundness of the PVS proof system, `metit` integrates MetiTarski and its RCF decision methods as trusted oracles into the PVS theorem prover.

Proving theorems over the reals with proof assistants such as PVS can require a significant amount of manual and computational effort. Sending difficult subproblems to trusted oracles is an accepted method for decreasing proof times. Since MetiTarski uses several external arithmetic decision methods (Mathematica, QEPCAD or Z3) itself for deciding the satisfiability of RCF sentences, the strategy `metit` greatly expands the number of options available to PVS for automatically dealing with problems from the theory of the reals. Experiments show that the new strategy is considerably better than other methods currently available to PVS for closing sequents containing real-valued functions.

For a certification environment, where external oracles may not be allowed, the PVS development includes several means to disable trusted strategies. For instance, `metit` has no effect on any theory that imports `MetiTarski@Disable`. Furthermore, the Emacs command `M-x disable-oracle MetiTarski` temporar-

Lemma	`interval` (s)	`metit` (s)	Speed up
sqrt23	1.39	0.154	9.27
sin6sqrt	1.76	0.120	14.67
sqrtx3	1.65	0.195	8.46
tr_35	1.97	0.680	2.77
tr_35_le	1.87	0.113	16.55
A_and_S	1.38	0.036	38.30
atan_implementation	2.55	0.154	16.56
ex1_ba	1.59	0.073	21.78
ex2_ba	1.51	0.049	30.82
ex3_ba	1.65	0.059	27.97
ex4_ba	1.71	0.078	21.92
ex5_ba	1.84	0.075	24.53
ex6_ba	1.60	0.105	15.24
ex7_ba	1.54	0.111	13.87

Table 1. Interval vs metit strategy run-times

ily disables the strategy during a PVS session and the `proveit` option `-disable MetiTarski` disables the strategy while reproving a PVS theory in batch mode.

References

1. Akbarpour, B., Paulson, L.C.: MetiTarski: An automatic theorem prover for real-valued special functions. Journal of Automated Reasoning 44, 175–205 (2010)
2. Daumas, M., Lester, D., Muñoz, C.: Verified real number calculations: A library for interval arithmetic. IEEE Transactions on Computers 58(2), 226–237 (February 2009)
3. Di Vito, B.: A PVS prover strategy package for common manipulations. Technical Memorandum NASA/TM-2002-211647, NASA Langley Research Center (2002)
4. Muñoz, C., Carreño, V., Dowek, G., Butler, R.: Formal verification of conflict detection algorithms. International Journal on Software Tools for Technology Transfer 4(3), 371–380 (2003)
5. Muñoz, C., Mayero, M.: Real automation in the field. Contractor Report NASA/CR-2001-211271, ICASE, Langley Research Center, Hampton VA 23681-2199, USA (December 2001)
6. Muñoz, C., Narkawicz, A.: Formalization of a representation of Bernstein polynomials and applications to global optimization. Journal of Automated Reasoning 51(2), 151–196 (2013), http://dx.doi.org/10.1007/s10817-012-9256-3
7. Narkawicz, A., Muñoz, C.: A formally verified generic branching algorithm for global optimization. In: Cohen, E., Rybalchenko, A. (eds.) Fifth Working Conference on Verified Software: Theories, Tools and Experiments (VSTTE 2013). Lecture Notes in Computer Science, vol. 8164, pp. 326–343. Springer (2014)
8. Owre, S., Rushby, J., Shankar, N.: PVS: A prototype verification system. In: Kapur, D. (ed.) 11th International Conference on Automated Deduction (CADE). Lecture Notes in Artificial Intelligence, vol. 607, pp. 748–752. Springer-Verlag, Saratoga, NY (Jun 1992)