

# Real Algebraic Strategies for MetiTarski Proofs



# Real Algebraic Strategies for MetiTarski Proofs

Grant Olney Passmore[1,2], Lawrence C. Paulson[1], and Leonardo de Moura[3]
gp351@cam.ac.uk, lp15@cam.ac.uk, leonardo@microsoft.com

[1] Computer Laboratory, University of Cambridge
[2] LFCS, University of Edinburgh
[3] Microsoft Research, Redmond

**Abstract.** MetiTarski [1] is an automatic theorem prover that can prove inequalities involving sin, cos, exp, ln, etc. During its proof search, it generates a series of subproblems in nonlinear polynomial real arithmetic which are reduced to true or false using a decision procedure for the theory of real closed fields (RCF). These calls are often a bottleneck: RCF is fundamentally infeasible. However, by studying these subproblems, we can design specialised variants of RCF decision procedures that run faster and improve MetiTarski's performance.

## 1 Introduction

MetiTarski [1] is an automatic theorem prover for special functions such as sin, cos, exp and ln, with variables ranging over the real numbers.[4] A typical problem is a universally quantified first-order formula involving inequalities between real-valued arithmetic expressions involving such functions; MetiTarski can prove many nontrivial problems in seconds, such as the following problem drawn from hybrid systems verification [6]:

$$\forall\, t \in (0,\infty), v \in (0,\infty)$$
$$((1.565 + 0.313\, v)\, \cos(1.16\, t) + (0.01340 + 0.00268\, v)\, \sin(1.16\, t))\, e^{-1.34\, t}$$
$$- (6.55 + 1.31\, v)\, e^{-0.318\, t} + v + 10 \geq 0$$

Internally, MetiTarski is a resolution theorem prover integrated with various decision procedures for the theory of real-closed fields (RCF) [2,7,9]. MetiTarski reduces its input problem to a series of logical combinations of polynomial inequalities, which are further reduced to true or false by an RCF decision procedure. Unfortunately, the RCF decision problem is hyper-exponential in the number of variables [3]. The RCF tests typically dominate the overall processor time, and thus far the success of our methods has been limited to problems in less than six variables. In this paper, we show that by analysing the structure of

[4] MetiTarski accepts first-order formulas over equations and inequalities, with terms involving real arithmetic, including an open-ended axiomatised collection of special functions. As this theory is undecidable, MetiTarski employs heuristic methods.

the RCF subproblems generated by MetiTarski, we can design specialised variants of RCF decision procedures. In many cases, RCF ceases to be a bottleneck, and MetiTarski's improved performance extends its practical reach.

MetiTarski requires axiom files that supply upper and lower bounds for the special functions of interest. In some cases, these bounds are polynomials, typically truncated Taylor series. More often, they are rational functions (fractions of polynomials) obtained from continued fraction approximations. MetiTarski includes arithmetic simplification designed to help transform special function inequalities so as to isolate one particular special function occurrence. The general resolution procedure, augmented with this simplification, automatically identifies relevant axioms, thereby replacing the special function occurrence by a polynomial or rational function that is an upper or lower bound, as appropriate for the context in which the special function occurs. In the case of a rational function, the division operator is eliminated through use of an axiom relating division with multiplication (again chosen by the general resolution mechanism). At this point, a special function inequality has been replaced by one or more polynomial inequalities.

The integration between resolution theorem proving and an RCF decision procedure takes the form of a novel simplification rule. Resolution, like DPLL-based SAT solving, is concerned with disjunctions of literals where a literal is an atomic formula or its negation. These disjunctions of literals are called *clauses*. When MetiTarski encounters a literal consisting of a polynomial inequality, it asks an RCF decision procedure whether this literal can possibly be true, taking into account its context. Formally, MetiTarski builds a conjunction combining all known clauses that express polynomial inequalities with the negations of the other literals in the clause. If this conjunction is logically inconsistent, then the literal is equivalent to false and can be deleted from the clause [1].

MetiTarski also uses RCF decisions to discard freshly-generated clauses that express nothing new, in the sense that their polynomial content is implied by other known polynomial inequalities. This RCF-based redundancy test is not essential, but it is a powerful heuristic nevertheless because it prevents the buildup of logically superfluous but syntactically complex facts.

The search for a proof typically generates hundreds of calls to the RCF decision procedure, often with gigantic formulas. In earlier work, we used QEPCAD-B [2] with a text-based interface, and in some cases formulas given to QEPCAD-B were longer than 50,000 characters. QEPCAD-B works extremely well for univariate problems, but it deteriorates rapidly with two or three variables. In such cases, when proofs are found, hardly any processor time spent in the resolution part of the proof search, and nearly all the time is spent in QEPCAD-B. A smarter approach to RCF will allow us to tackle harder problems, and to improve the speed at which we solve problems. We describe an approach to such improvements below.

### 1.1 Motivating Hypotheses

The following hypotheses motivate our work:

1. By studying the structure of the sequences of RCF subproblems MetiTarski generates during its proof search, we can devise specialised RCF proof methods which outperform general "off the shelf" RCF proof methods on these sequences of RCF subproblems.
2. By making use of these specialised RCF proof methods during MetiTarski's proof search, we can significantly improve MetiTarski's performance.

The results in this paper strongly support these hypotheses. Moreover, extrapolating from the particular case of MetiTarski, we believe this work supports a broader hypothesis: That a methodology of studying the structure of generated subproblems and specialising decision methods to them can lead to substantial gains for many similarly arranged combinations of automatic proof methods.

### 1.2 Overview of Contributions

Based upon detailed analysis of the RCF subproblems generated by MetiTarski, we have made the following improvements:

1. A method for quickly recognising the satisfiability of generated $\exists$ RCF subproblems through retaining, during any given MetiTarski run, a history of past models produced for satisfiable RCF subproblems. This improvement works because $\exists$ RCF subproblems generated during MetiTarski proof search very often have models in common with each other. To instrument this improvement, we communicate models between MetiTarski and the external RCF decision methods it invokes. When a retained past model consists only of rational points, we test the model against new $\exists$ RCF subproblems from within MetiTarski alone.
2. We observe that the univariate polynomials appearing in RCF subproblems generated by MetiTarski are almost always irreducible over $\mathbb{Z}[x]$. Thus, attempting to factor them, which is a step applied by default by most RCF decision methods known to us, is almost always a waste of time. For the $\exists$ RCF decision procedure in the SMT solver Z3 [4], we observe that per RCF instance, disabling univariate factorisation has only a small speed-up, usually less than 0.02 seconds.[5] However, for typical univariate RCF subproblems, this speed-up is anywhere from 40% - 90% of the total decision method run time. As MetiTarski may generate many thousands of RCF subproblems during its search for a single proof, each of which may contain tens of different polynomials, this speed-up nontrivially aggregates, leading to serious gains.

Methodologically, these improvements were motivated by extensive computational study of the RCF subproblems generated during MetiTarski proofs.[6]

[5] The Z3 program we use in this paper is a new unreleased version with `nlsat`, a novel approach to making $\exists$ RCF decisions. It (and an accompanying paper [7]) may be retrieved: `http://cs.nyu.edu/~dejan/nonlinear/`.

[6] This collection of RCF subproblems consists of over 400,000 $\exists$ RCF sentences, each occurring in MetiTarski's search for a proof of one of $\approx 800$ special function inequality benchmarks drawn from many mathematical, scientific and engineering sources.

The success of these methods is supported by extensive experimentation as well. As we shall see, their combination allows MetiTarski to find proofs much more quickly than it can with non-specialised "off the shelf" RCF proof methods.

## 2 Model Sharing

Given $\varphi$, a universally quantified boolean combination of special function inequalities, MetiTarski attempts to prove $\varphi$ through a combination of resolution and RCF reasoning. For the MetiTarski problems considered in this paper, these generated RCF subproblems are always purely $\exists$. We will say an $\exists$ RCF sentence $F$ is *satisfiable* if it is true, i.e., if $\exists \boldsymbol{r} \in \mathbb{R}^n$ s.t. $QF(F)(\boldsymbol{r})$ holds, where $QF(F)$ is the quantifier-free matrix of $F$. We say $F$ is *unsatisfiable* otherwise. Let $F_1, \ldots, F_k$ be the sequence of RCF subproblems generated by MetiTarski during its search for a proof of $\varphi$. Then, the following hold:

1. $F_i$ only contributes to a MetiTarski proof when $F_i$ is unsatisfiable over $\mathbb{R}^n$,
2. Many of the $F_i$ share common subexpressions with each other.

From the first point, we see that time spent analysing the truth of satisfiable RCF subproblems is ultimately time wasted for MetiTarski. Thus, it is desirable to have methods for quickly recognising and throwing away satisfiable $F_i$. Combining this desire with the second point above, we are led naturally to the following question:

> Given a satisfiable RCF subproblem $F_i$ and a subsequent satisfiable RCF subproblem $F_{i+k}$, is it often the case that $F_i$ and $F_{i+k}$ have a model in common?

As we will see, the answer to this question is a resounding *yes*. These observations lead to one of our key improvements to MetiTarski: By recording in MetiTarski models that an RCF decision procedure has found for satisfiable $F_i$'s, we can gain a tremendous speed-up by using these past models to quickly recognise the satisfiability of subsequently generated RCF subproblems. The overhead involved in communicating, storing and testing these models is far outweighed by the savings made through avoiding invoking an RCF decision method.

### 2.1 MetiTarski Proof Search in More Detail

To motivate this model-sharing improvement to MetiTarski, let us study the sequence of RCF subproblems generated during MetiTarski's search for a proof of a particularly simple special function inequality.[7] In our benchmark set, this problem is named `max-sin-2` and is the following claim over $\mathbb{R}$:

[7] This problem can in fact be solved quickly by Mathematica directly, using some recent methods it contains for computing with transcendental functions [10]. We focus on this problem in our discussion of MetiTarski's proof search for didactic reasons.

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

In searching for a proof of this theorem, MetiTarski will make use of axioms it knows for sin, cos and max. With default settings, and without using any of the enhancements we describe in this paper, MetiTarski finds a proof consisting of 600 steps. Each step is either a resolution step, a substitution step, an arithmetical simplification step, or an RCF decision step. This proof makes use of three different lower bounds and three different upper bounds for cos, six different upper bounds and six different lower bounds for sin, and two definitional axioms for max. For example, one of the sin lower bounds used is the following:[8]

```
cnf(sin_lower_bound_5_neg, axiom,
    (0 < X |
     ~lgen(R, Y,
           X - 1/6 * X ^ 3 + 1/120 * X ^ 5 - 1/5040 * X ^ 7 +
           1/362880 * X ^ 9 - 1/39916800 * X ^ 11 +
           1/6227020800 * X ^ 13 - 1/1307674368000 * X ^ 15 +
           1/355687428096000 * X ^ 17 - 1/121645100408832000 * X ^ 19
           + 1/51090942171709440000 * X ^ 21) | lgen(R, Y, sin(X)))).
```

Many of the intermediate clauses used in the proof contain very large polynomials with high coefficient bit-width and degree. When pretty-printed to a text file at 75 columns per line, this proof consists of 12,453 lines.

Let us now examine some properties of MetiTarski's search for this proof. The total number of RCF inferences used in the proof is 62. But how many RCF subproblems were generated and sent to an RCF decision procedure in search of this proof? This number is much higher: 2,776. Of these subproblems, 2,221 are satisfiable. Thus, over 80% of the RCF subproblems generated cannot contribute anything towards MetiTarski's proof. Deciding their satisfiability is a waste of time. This waste can be large, as the RCF subproblems are often very complicated. For instance, the set of all polynomials appearing in these 2,776 RCF subproblems has the following statistics: max total degree is 24, average total degree is 3.53, max coefficient bit-width is 103, and average coefficient bit-width is 21.03.

To get an idea of the expense involved in deciding these satisfiable RCF subproblems generated by MetiTarski, let us examine them using Mathematica's `Reduce` command. This command is one of the best and most sophisticated general-purpose tools for deciding RCF sentences, containing highly-tuned implementations of a vast array of approaches to making RCF decisions [8,9].

Using Mathematica's `Reduce` to decide all of these 2,776 RCF sentences, we see that 253.33 seconds is spent in total. Of that time, 185.28 seconds is spent deciding the satisfiable formulas. Thus, over 70% of the total RCF time for MetiTarski's proof search is spent deciding formulas which in the end can contribute

[8] Here `lgen(R, X, Y)` is a generalised inequality relation. It eliminates the need to have separate instances of the axiom for $<$ and $\leq$.

nothing to MetiTarski's proof. Such results are typical. Table 1 analyses ten representative problems. For each, it displays the effort (in terms of the number of RCF problems and the time taken deciding them), followed by the subset of this effort that is wasted on satisfiable problems and finally the percentage of wasted effort, again in terms of the number of problems and the time taken. We list the contents of these problems in Table 2. Clearly, quick methods for identifying and discarding satisfiable RCF subproblems could greatly improve performance.

**Table 1.** RCF Subproblem Analysis for Ten Typical Benchmarks

| Problem | All RCF | | SAT RCF | | % SAT | |
|---|---|---|---|---|---|---|
| | # | secs | # | secs | # | secs |
| CONVOI2-sincos | 268 | 3.28 | 194 | 2.58 | 72% | 79% |
| exp-problem-9 | 1213 | 6.25 | 731 | 4.11 | 60% | 66% |
| log-fun-ineq-e-weak | 496 | 31.50 | 323 | 20.60 | 65% | 65% |
| max-sin-2 | 2776 | 253.33 | 2,221 | 185.28 | 80% | 73% |
| sin-3425b | 118 | 39.28 | 72 | 14.71 | 61% | 37% |
| sqrt-problem-13-sqrt3 | 2031 | 22.90 | 1403 | 17.09 | 69% | 75% |
| tan-1-1var-weak | 817 | 19.5 | 458 | 7.60 | 56% | 39% |
| trig-squared3 | 742 | 32.92 | 549 | 20.66 | 74% | 63% |
| trig-squared4 | 847 | 45.29 | 637 | 20.78 | 75% | 46% |
| trigpoly-3514-2 | 1070 | 17.66 | 934 | 14.85 | 87% | 84% |

Now, given our previous discussions, it is natural to ask the following: How many of these satisfiable RCF subproblems share models with each other? Obtaining an exact answer to this question is certainly computationally infeasible. However, we can obtain a lower bound. We will do this in the following simple way: Whenever the RCF procedure decides a formula to be satisfiable, we will ask it to report to us a model satisfying the formula, and we will store this model within a *model history* data-structure in MetiTarski. Note that these models may in general contain irrational real algebraic points. Whenever we encounter a new RCF subproblem, we will first check, within MetiTarski, whether this RCF subproblem is satisfied by any rational model we have recorded within the model history.

Performing this experiment, we see that at least 2,172 of the 2,221 satisfiable RCF subproblems share a common model with a previously generated SAT RCF subproblem. Moreover, only 37 separate rational models were used to satisfy all of these 2,172 formulas. Note that these numbers are very much *lower* bounds, as we (i) only consider the particular models previously recorded (i.e., perhaps $F_i$ and $F_{i+k}$ share a model, but this common model is different than the one we have recorded for $F_i$), and (ii) we have only considered common rational models.

In Table 3, we show this type of model sharing analysis for the same collection of ten benchmark problems encountered previously. For each MetiTarski prob-

**Table 2.** Typical MetiTarski Benchmarks

| |
|---|
| `CONVOI2-sincos`<br>$\forall\, t \in (0,\infty), v \in (0,\infty)$<br>$((1.565 + 0.313\, v)\, \cos(1.16\, t) + (0.01340 + 0.00268\, v)\, \sin(1.16\, t))\, e^{-1.34\, t}$<br>$-\, (6.55 + 1.31\, v)\, e^{-0.318\, t} + v + 10 \geq 0$ |
| `exp-problem-9`<br>$\forall\, x \in (0,\infty)$<br>$\dfrac{1 - e^{-2\, x}}{2\, x\, (1 - e^{-x})^2} - \dfrac{1}{x^2} \leq \dfrac{1}{12}$ |
| `log-fun-ineq-e-weak`<br>$\forall\, x \in (0,12), y \in (-\infty,\infty)$<br>$xy \leq \dfrac{1}{5} + x\, \ln(x) + e^{y-1}$ |
| `max-sin-2`<br>$\forall\, x \in (-8,5)$<br>$\max(\sin(x), \sin(x+4), \cos(x)) > 0$ |
| `sin-3425b`<br>$\forall x \in (0,\infty), y \in (-\infty,\infty)$<br>$(x < y \wedge y^2 < 6) \Rightarrow \dfrac{\sin(y)}{\sin(x)} \leq 10^{-4} + \dfrac{y - \frac{1}{6}y^3 + \frac{1}{120}y^5}{x - \frac{1}{6}x^3 + \frac{1}{120}x^5}$ |
| `sqrt-problem-13-sqrt3`<br>$\forall x \in (0,1)$<br>$1.914 \dfrac{\sqrt{1+x} - \sqrt{1-x}}{4 + \sqrt{1+x} + \sqrt{1-x}} \leq 0.01 + \dfrac{x}{2 + \sqrt{1-x^2}}$ |
| `tan-1-1var-weak`<br>$\forall x \in (0,1.25)$<br>$\tan(x)^2 \leq 1.75\, 10^{-7} + \tan(1) \tan(x^2)$ |
| `trig-squared3`<br>$\forall x \in (-1,1), y \in (-1,1)$<br>$\cos(x)^2 - \cos(y)^2 \leq -\sin(x+y) \sin(x-y) + 0.25$ |
| `trig-squared4`<br>$\forall x \in (-1,1), y \in (-1,1)$<br>$\cos(x)^2 - \cos(y)^2 \geq -\sin(x+y) \sin(x-y) - 0.25$ |
| `trigpoly-3514-2`<br>$\forall x \in (-\pi,\pi)$<br>$2\, \lvert \sin(x) \rvert + \lvert \sin(2\, x) \rvert \leq \dfrac{9}{\pi}$ |

lem considered, we show (i) the number of SAT RCF subproblems generated, (ii) the number of those problems which could be recognised to be SAT using the simple rational model-sharing described above, (iii) the number of different ra-

**Table 3.** Model Sharing Lower Bounds for Ten Typical Benchmarks

| Problem | # SAT | # SAT by MS | # $\mathbb{Q}$ Models | # Successful |
|---|---|---|---|---|
| `CONVOI2-sincos` | 194 | 168 | 9 | 7 |
| `exp-problem-9` | 731 | 720 | 11 | 7 |
| `log-fun-ineq-e-weak` | 323 | 305 | 24 | 18 |
| `max-sin-2` | 2,221 | 2,172 | 37 | 37 |
| `sin-3425b` | 72 | 64 | 8 | 6 |
| `sqrt-problem-13-sqrt3` | 1403 | 1350 | 26 | 21 |
| `tan-1-1var-weak` | 458 | 445 | 13 | 9 |
| `trig-squared3` | 549 | 280 | 15 | 11 |
| `trig-squared4` | 637 | 497 | 21 | 16 |
| `trigpoly-3514-2` | 934 | 4 | 4 | 2 |

tional models stored in MetiTarski's model history, and (iv) the number of those models which were successfully shared between at least two RCF subproblems (the *successful* models in the model history). We see that with the exception of `trigpoly-3514-2`, a very large majority of the SAT RCF subproblems can be recognised to be satisfiable through the use of past rational models. We have found the vast majority of our benchmark problems to exhibit behaviour consistent with the first nine problems in the table. We note that of those problems considered, `trigpoly-3514-2` is the only one involving $\pi$, which is approximated by MetiTarski using rational upper and lower bounds. Perhaps the presence of $\pi$ in the problem has something to do with why its rational model sharing lower bounds are so much lower than the rest.

Clearly, there is much potential for improving MetiTarski through using past models of SAT RCF subproblems to quickly recognise subsequent SAT RCF subproblems. However, we have found that in some cases, the cost of finding a suitable model in our model history can be quite high. This is due to the fact that evaluating very large RCF formulas upon rational numbers of very large bit-width can become expensive (even if somewhat sophisticated approaches to polynomial sign determination are employed).

To efficiently apply this model-sharing technique in the context of MetiTarski's proof search, we have found it necessary to seek some heuristic methods for prioritising the models based upon their success rates in recognising SAT RCF subproblems. Through experimentation, we have found that prioritising models based upon *recent success* to be most useful. We store all rational models within MetiTarski, but maintain at any time a list of the ten most successful models, ordered descendingly by how recently they have been successfully applied to recognise a SAT RCF subproblem. When a new RCF subproblem is encountered, we first try the prioritised models in order. If that fails, then we try the remaining models in our model history, this time in an order based solely upon success rate.

## 3 Univariate Factorisations

RCF decision procedures typically devote a significant effort to factoring polynomials, effort that is wasted if a polynomial is irreducible. In our case, it has turned out that most of the polynomials generated by MetiTarski are irreducible. This is presumably because most of the polynomials we use to bound special functions are themselves irreducible. Frequently, a bound is the ratio of two polynomials; MetiTarski will then multiply both sides by the denominator. The resulting simplifications do not necessarily have to yield another irreducible polynomial; empirically, however, this usually happens.

Of the well-known transcendental functions, polynomials involved in their bounds used by MetiTarski only have very simple factors, if they have any at all. In the case of the functions $\sin(X)$ and $\tan^{-1}(X)$, this factor is simply $X$, which is unsurprising because their value is zero when $X = 0$. Similarly, for the function $\ln(X)$, some polynomials have $X - 1$ as a factor. On the other hand, bounds for the function $\mathrm{sqrt}(X)$ have many non-trivial factors. Note that the square root bounds are derived using Newton's method, while most other bounds come from Taylor series or continued fractions.

**Table 4.** Factorisation in RCF Subproblems for Typical Univariate Benchmarks

| Problem | # Factor | # Irreducible | % Runtime |
|---|---|---|---|
| `asin-8-sqrt2` | 7791 | 5975 (76.7%) | 22.4% |
| `atan-problem-2-sqrt-weakest21` | 65304 | 63522 (97.3%) | 55.4% |
| `atan-problem-2-weakest21` | 9882 | 8552 (86.5%) | 2.2% |
| `cbrt-problem-5a` | 88986 | 61068 (68.6%) | 38.6% |
| `cbrt-problem-5b-weak` | 138861 | 25107 (18.0%) | 53.1% |
| `cos-3411-a-weak` | 150354 | 138592 (92.1%) | 53.9% |
| `ellipse-check-2-weak2` | 5236 | 3740 (71.4%) | 88.7% |
| `ellipse-check-3-ln` | 1724 | 1284 (74.4%) | 86.7% |
| `ellipse-check-3-weak` | 12722 | 9464 (74.3%) | 77.9% |

Table 4 analyses a representative set of MetiTarski problems. For each, it displays the number of times the factorisation subprocedure is invoked in Z3, the number of times the polynomial argument is irreducible, the percentage of irreducible polynomials, and the percentage of runtime spent in the factorisation subprocedure.[9]

For univariate benchmarks, we observed that the overhead of polynomial factorisation is quite significant. Moreover, our RCF procedure in Z3 does not seem to benefit from factorisation as a preprocessing step even when polynomials can be factored. Consider the problem instances `ellipse-check-2-weak2` and

[9] These experiments were performed on an Intel Core i7-2620M 2.70GHz with 8GB RAM running Windows 7 64-bit.

`ellipse-check-3-weak` from Table 4. MetiTarski creates respectively 803 and 1569 RCF subproblems for these instances. The RCF procedure in Z3 spends respectively 88.69% and 77.95% of the runtime in the polynomial factorisation subprocedure. Although each instance can be solved in less than 20 milliseconds, a signficant amount of time can be saved by disabling the factorisation subprocedure. The experimental results in Sect. 4 demonstrate that this indeed the case.

## 4 Experimental Results

We have compared four separate MetiTarski runs using different RCF decision procedures: QEPCAD, Mathematica, Z3 and finally our specially modified version of Z3 incorporating the reduced factorisation strategy (cf. Sect. 3) and prioritised model-sharing (cf. Sect. 2).[10] We have allowed up to 120 seconds per problem, using a Perl script to count how many theorems were proved in 10, 20, ..., 120 seconds processor time (the total of the time spent in proof search and RCF calls). These experiments used a subset of 409 problems taken from our full set of 853 problems. This subset omits trivial problems (defined as those that can be proved in less than one second). It also omits the existential problems, of which there are 39, because none of the new methods work for them.[11] Figure 1 displays our results:[12]

For runtimes up to about 60 seconds, the graphs show a clear advantage for Z3 as modified using Strategy 1, but even unmodified Z3 does very well. By 120 seconds, all four runs appear to converge. This conclusion is not quite accurate, as the different decision procedures are succeeding on different problems. Mathematica does particularly well on problems with three or more variables. QEPCAD cannot prove many of these, but it does very well on univariate problems. As more processor time is allowed, Mathematica is able to prove more theorems that only it can prove, giving it an advantage.

We also compared the four decision procedures in terms of the number of problems for which they find the fastest proof. We use a threshold in this comparison, counting a proof only if it is faster by a given margin (10%, 50% or 100%, respectively) than all other proofs found; these results appear in Figure 2.

With a threshold of 10% faster, Z3 modified by Strategy 1 dramatically outperforms all other decision procedures. Its advantage decreases rapidly as this threshold is increased, while Mathematica's score largely holds steady. The situation is complicated by unique proofs: 18 theorems are proved by one system

[10] These experiments were performed on a 2×2.4 GHz Quad-Core Intel Xeon PowerMac with 10GB of 1066 MHz DDR3 RAM using QEPCAD-B 1.62, Mathematica 8.0.1 and Z3 4.0. This same machine and Mathematica installation were used for the experiments in Sect. 2.

[11] The extension to MetiTarski allowing existentially-quantified problems must be seen as experimental. It only works on trivial problems such as $\forall y\, \exists x\, \sinh x > y$.

[12] There is a web resource for this paper containing the MetiTarski source code, benchmarks and related data: `http://www.cl.cam.ac.uk/~gp351/cicm2012/`.

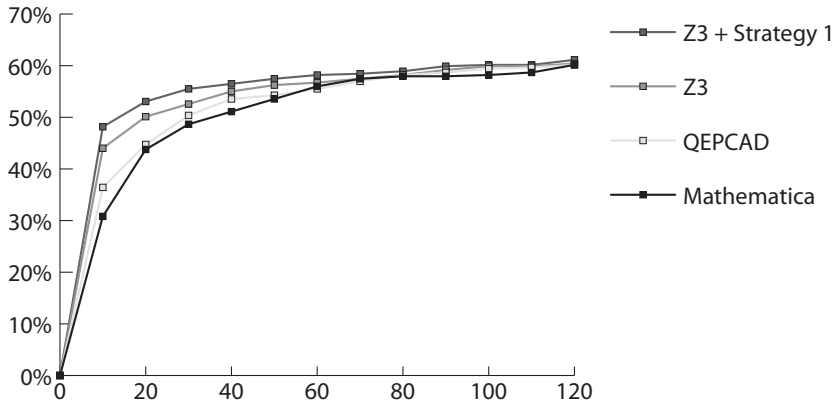


**Fig. 1.** Theorems Proved (by percentage of the total)

only, and of these, Mathematica proves 15. (QEPCAD-B proves one, while modified Z3 proves two.) Mathematica's superiority for higher-dimensional problems (each theorem that it uniquely proves has at least two variables, generally more) gives it an advantage as the threshold is increased, because a unique proof will always be counted as the fastest. If the threshold is pushed high enough, only unique proofs will be counted, and here Mathematica has an inbuilt advantage. Modified Z3 remains top even with the threshold of 200% faster (which means three times faster). Mathematica finally wins at four times faster, with 17 problems against 8 for modified Z3, but these are mostly unique proofs rather than faster proofs.

Our data suggest another question: how is it that QEPCAD-B so often outperforms Mathematica, especially on univariate problems? Mathematica has much better algorithms for real algebraic numbers, and is generally more up-to-date. Overheads outside of Mathematica's core RCF decision procedure are presumably to blame. At present, we do not know whether these overheads are concerned with parsing, preprocessing or something else altogether.

## 5 Future Work

We see many ways this work might be improved and extended. First, we would like to better understand how the lineage of RCF subproblems (i.e., the clauses from which the RCF subproblems were generated) influences model sharing. If two SAT RCF subproblems share a common ancestry, is it more likely that they might share a model? This seems likely. It seems plausible that lineage-based methods for prioritising which stored models we should try may yield serious efficiency improvements. It would also be very interesting to incorporate machine learning into this process. Second, we would like to make use of irrational real algebraic models in our model-sharing machinery. Currently, only rational models are used to recognise SAT RCF subproblems from within MetiTarski.

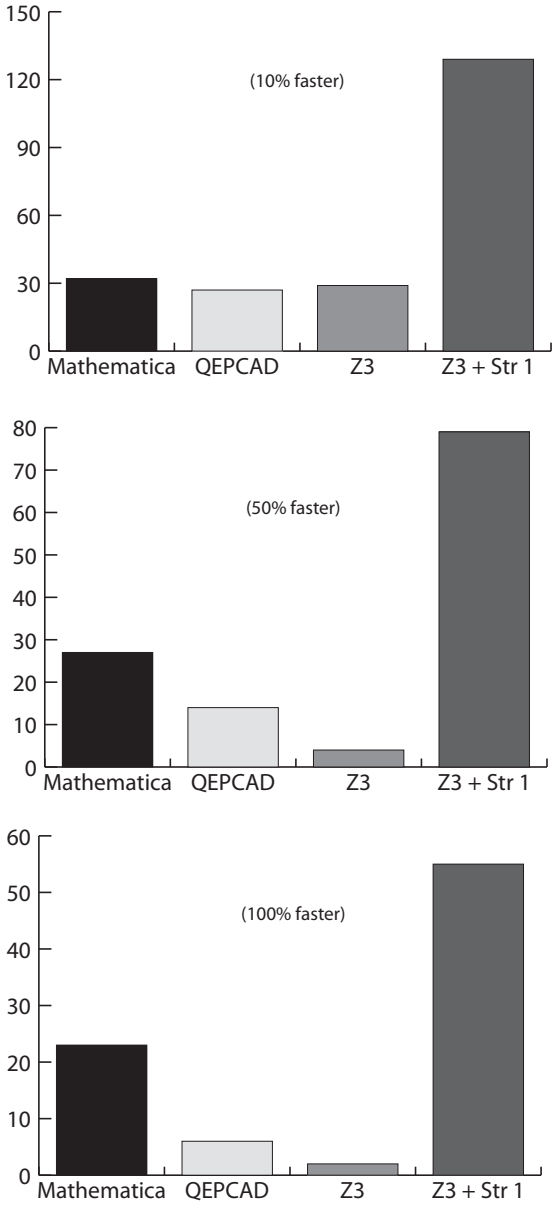


**Fig. 2.** Number of Fastest Proofs (by the Given Threshold) Per Run

One approach which interests us involves using retained real algebraic models to guide an RCF proof procedure towards certain regions of $\mathbb{R}^n$. This may involve combining techniques based upon interval constraint propagation and *paving* [5] to guide the manner in which Z3 explores its search space, for instance.

## 6 Conclusion

We have shown that through detailed analysis of the RCF subproblems generated during MetiTarski's proof search, we can devise specialised variants of RCF decision procedures that greatly outperform general-purpose methods on these problems.

The approach described here is applicable to the design of any expensive proof procedure. Given a sufficiently large corpus of representative problems, the general-purpose procedure can be tuned, which should yield dramatically better results. This principle also applies when proof procedures are combined: the subsidiary proof engine should not be viewed as a black box, but should be refined by analysing the generated problems given to it. It follows that expensive proof procedures should offer easy customisation so that their users can try such refinements with the least effort.

*Acknowledgements.* The research was supported by the Engineering and Physical Sciences Research Council [grant numbers EP/I011005/1 and EP/I010335/1]. We thank the referees for their helpful suggestions which improved our paper.

## References

1. B. Akbarpour and L. Paulson. MetiTarski: An Automatic Theorem Prover for Real-Valued Special Functions. *Journal of Automated Reasoning*, 44(3):175–205, Mar. 2010.
2. C. W. Brown. QEPCAD-B: A System for Computing with Semi-algebraic Sets via Cylindrical Algebraic Decomposition. *SIGSAM Bull.*, 38:23–24, March 2004.
3. J. H. Davenport and J. Heintz. Real Quantifier Elimination is Doubly Exponential. *J. Symb. Comput.*, 5:29–35, 1988.
4. L. de Moura and N. Björner. Z3: An efficient SMT solver. In *TACAS'08*, 2008.
5. L. Granvilliers and F. Benhamou. RealPaver: An Interval Solver using Constraint Satisfaction Techniques. *ACM Trans. on Mathematical Software*, 32:138–156, 2006.
6. S. Gulwani and A. Tiwari. Constraint-based approach for analysis of hybrid systems. In A. Gupta and S. Malik, editors, *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 190–203. Springer, 2008.
7. D. Jovanovič and L. de Moura. Solving Nonlinear Arithmetic. In *IJCAR'12*, 2012.
8. A. Strzebonski. Solving Systems of Strict Polynomial Inequalities. *Journal of Symbolic Computation*, 29(3):471 – 480, 2000.
9. A. Strzebonski. Cylindrical Algebraic Decomposition using Validated Numerics. *Journal of Symbolic Computation*, 41(9):1021 – 1038, 2006.
10. A. Strzebonski. Real Root Isolation for Tame Elementary Functions. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, ISSAC '09, pages 341–350, New York, NY, USA, 2009. ACM.