MetiTarski User Guide

Lawrence C Paulson
University of Cambridge

21 October 2014

1 Introduction

MetiTarski is an automatic theorem prover based on a combination of resolution and a decision procedure for the theory of real closed fields. It is designed to prove theorems involving real-valued special functions such as log, exp, sin, cos and sqrt. In particular, it is designed to prove universally quantified inequalities involving such functions. This problem is undecidable, so MetiTarski is necessarily incomplete. Nevertheless, MetiTarski is remarkably powerful. Here are a few of the hundreds of theorems that it can prove, automatically and in seconds.

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

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

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

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

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

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

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

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

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

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

Work on MetiTarski started in 2006, jointly with Behzad Akbarpour. The original manual experiments [4] were later automated with the help of a basic decision procedure [1]. Crucial is the use of upper and lower bounds for the special functions of interest. By 2008, MetiTarski had a basic set of such bounds, largely based on Taylor expansions, and using QEPCAD as its decision procedure [2]. The next major milestone was the introduction of upper and lower bounds based on continued fractions [3]. Until this stage, MetiTarski's underlying proof engine was based on standard resolution. The introduction of case-splitting (originally in simulated form, later superseded by James Bridge's implementation of backtracking) provided decisive additional power for many problems [6]. Other work, chiefly by Grant Passmore, extended the range of decision procedures to include Mathematica and Z3; later, Z3 with a specialised "strategy 1" became the default configuration [12]. Passmore implemented an interval constraint solver and many other refinements.

The development of MetiTarski went hand-in-hand with a variety of applications, chiefly at Concordia University (in Montréal) and Cambridge. These applications chiefly concerned hybrid systems [5] and analogue circuits [8, 11]. William Denman has continued to explore applications mostly in the aerospace domain [7, 9, 10].

For definitive and comprehensive about MetiTarski, consult the journal articles [6, 3]. For more casual overviews and speculations about future research, some of Paulson's invited lectures [14, 15] are worth reading. Links to most of these papers can be found on the MetiTarski web page, `www.cl.cam.ac.uk/~lp15/papers/Arith/`.

2 Compiling and installing MetiTarski

Before compiling and installing MetiTarski, first install Poly/ML (from `www.polyml.org`) and at least one decision procedure as outlined below.

2.1 Installing a decision procedure

MetiTarski requires an external algebraic decision method (EADM). MetiTarski supports three such tools. For each, certain *environment variables* must be set up to locate them.

QEPCAD B version 1.69 or later is available from `www.usna.edu/CS/~qepcad/B/QEPCAD.html` in source form. The environment variable `qe` must point to the qesource subdirectory of the QEPCAD B installation, so that `$qe/bin/qepcad` is the compiled QEPCAD B binary.

Mathematica version 8.0 or later is a commercial product. The environment variable `MATH_KERNEL` must point to the binary for the console-mode Mathematica interface. This file is called `math` on Linux and `MathKernel.exe` on Windows. On Mac OS X, it must point into the Mathematica application itself:

```
/Applications/Mathematica.app/Contents/MacOS/MathKernel
```

Remark: we have recently encountered the following error message when attempting to use Mathematica under Mac OS X:

```
dlopen(…/SystemFiles, 1): image not found
```

A workaround is to create the missing file as a symbolic link to

```
/Applications/Mathematica.app/SystemFiles .
```

Z3 version 4.0 or later can be obtained from `z3.codeplex.com/`. The environment variable `Z3_NONLIN` must point to the Z3 binary.

The default is Z3, which is available in binary form and is free to non-commercial users. QEPCAD is also free, but must be built from sources. QEPCAD is best for univariate problems. Mathematica is available in many institutions under a site license, and is very good for problems involving three or four variables.

2.2 Building MetiTarski

To build MetiTarski, first install Poly/ML version 5.5 or later. For improved performance, you can build Poly/ML using the GMP multiple-precision arithmetic package: first install GMP (from `gmplib.org`) and then build Poly/ML including the line

```
./configure --with-gmp=yes
```

You may need to set the variables `LDFLAGS`, `CFLAGS` and `CXXFLAGS` to locate `gmp`.

To compile and install MetiTarski in its default location of `/usr/local/bin`, enter the following three commands:

```
./configure
make
sudo make install
```

To install MetiTarski in a different directory, use the `--prefix` option with the `configure` command. For example, to install it in subdirectory `bin` of your home directory, use

```
./configure --prefix=$HOME
```

NB the prefix should *not* end with "`bin`", as this will be added automatically.

3 Axiom and problem files

Before preparing your own problems, please look at our many examples. They will give you an impression of the sort of problem MetiTarski can solve. MetiTarski can include relevant axiom files automatically (using the option `--autoInclude`). Better results can be obtained if you select the axiom files yourself, omitting those that aren't strictly necessary. But this requires some experience and skill.

Directory `tptp` contains many axiom and problem files. The problem files use the `include` directive to insert axioms relevant to the functions mentioned in those problems. Here is an example:

```
include('Axioms/general.ax').
```

The pathname mentioned by such a directive is looked up relative to a base directory, which can be specified to MetiTarski using the `--tptp` command line option. If that option is not used, the base directory is obtained from the current setting of the `TPTP` environment variable. However, if a pathname starts with the "`/`" character, then it is regarded as absolute and no base directory is used. If an included file is not found, then MetiTarski will terminate with an error message.

To run MetiTarski on an individual problem, use the binary executable, metit. here is an example:

```
metit --verbose 0 --show proof --tptp .. atan-problem-15.tptp
```

Here we see three options being given. The first suppresses all prover output apart from the final result. The second requests the prover to display the final proof. The default verbosity is 1, whilst 2 and 3 present more detail. The third option specifies the TPTP directory. The last item on the command line is the filename containing the problem; you may put a series of file names here and the problems will be attempted sequentially.

Version 1.4 introduced experimental, limited support for problems involving existential quantification. In the supplied set of problems, the existential ones have names beginning with the letter X. Suggestions for more interesting problems are welcome! These cannot be solved using Z3 as the EADM.

4 Defining your own functions

Non-recursive functions (effectively, abbreviations) can be defined using axioms such as the following:

```
cnf(f, axiom, (f(X) = 2*(X - X^3))).
```

An example of using such functions is the problem `ellipse-check-2-fun.tptp`. More examples are available, in files with names matching the pattern `*-fun.tptp`. Functions may refer to one another, but it is the user's responsibility to avoid recursive definitions, which are *not* checked for termination.

5 Command line options

5.1 Options to specify the EADM

The default algebraic decision method is Z3 with "Strategy 1" [12].

To specify a different decision method, use one of the following options:

`--z3` Use plain Z3 (instead of the default Strategy 1)

`-m` or `--mathematica` Use Mathematica as the external algebraic decision method.

`--qepcad` Use QEPCAD as the external algebraic decision method.

`--nsatz_eadm` Use MetiTarski's built-in decision method (combining ICP with a search for real nullstellensatz witnesses) before calling the external one.

One of the options `--z3`, `--mathematica` or `--qepcad` must be given. A good combination is `--nsatz_eadm -m`. This is an implementation of Passmore's ICP-enhanced Tiwari calculus [13, §6.3.3]

Other command-line options control MetiTarski's operation and heuristics. The following options are the most important. The backtracking and case-splitting settings can make the difference between success and failure, and unfortunately there is no obvious way to choose the best ones for a given problem. Most of the numerical parameters below are only for advanced users.

5.2 Basic options

`-p` or `--show proof` A proof (if found) will be produced on the standard output.

`--time` $\langle float \rangle$ limits the processor time used in the proof attempt (default 600 seconds); decimals are allowed, e.g., 0.1 for 100 milliseconds

`--autoInclude`, `--autoIncludeExtended`, `--autoIncludeSuperExtended` includes axiom files automatically (with extended or extra extended accuracy, respectively)

`-t` or `--verbose` $\langle 0 \ldots 4 \rangle$ specifies the degree of verbosity for the proof search The default is 1, which displays information about CPU usage: a full stop (`.`) for every 10 seconds of Metis time and a (`+`) for every 10 secs of RCF time.

5.3 More advanced heuristic settings

* Indicates the default alternative.

`--backtracking off / on*` disables/enables backtracking.

`--cases off` switches off case splitting.

`--cases` m If backtracking is on, then m is the maximum size of the split stack (default 100); if backtracking is off, then m is the maximum number of splits (default 3000).

`--cases` m`+`n m is as above, while n sets the weight factor for non-SOS clauses in tenths. Thus a value of n of 10 is neutral (the factor is 1.0 and the weight is unchanged).

`--proj_ord off / on*` enables/disables automatic selection of a projection ordering for QEPCAD, which can have a great impact on multivariate problems.

`--maxalg` $\langle n \rangle$ sets the maximum size of an algebraic clause to be retained as part of the context in QEPCAD calls. A value of 50 or 75 may benefit some problems that have two or three variables, while a few problems require at least 500. The default is 100.

`--rerun off / on*` controls whether to try again (with maxalg = 1001) after running out of clauses, instead of just giving up.

`--paramodulation off / on*` disables/enables the paramodulation rule.

`--maxweight` $\langle w \rangle$ sets the maximum weight of a retained clause, default 1000 * (.5 + .5 * SOS_factor). Smaller values save memory but may cause MetiTarski to quit prematurely because it has run out of clauses. The maximum weight observed in any proof is 1007.

`--tptp` $\langle d \rangle$ specifies the TPTP installation directory

`--tstp` generates standard TSTP proofs (no infixes, etc.), for use with TSTP analysis tools

`--full` includes variable instantiations in proofs

`-q` **or** `--quiet` no output: indicates provability with return value

6 Perl scripts

Directory scripts contains Perl scripts that are useful for generating and running problem sets: runmetit and addaxioms.

MetiTarski can be given a list of problem files on the command line, but the Perl script `runmetit.pl` provides more flexibility. It is especially useful when many problems are to be attempted, each for a limited time. A log file summarising the outcomes for all the problems will be stored in a file entitled `STATUS-Metit-`*yyyy-mm-dd*. Here, we use runmetit to run MetiTarski on a directory of problems.

```
runmetit.pl --time 10 --proofs --tptp $HOME/metit-2.4/tptp
```

Here is a summary of the main options. Others are documented in the script itself.

`--time` limits the processor time in seconds. It is not especially accurate.

`--proofs` produces proofs as new files, in a directory entitled Proofs-Metit-yyyy-mm-dd.

`--threads` **(default 2)** specifies the maximum number of threads to be used in parallel. (Too many threads may crash your machine! Your Mathematica licence may forbid multiple threads.)

`--options "`*opts*`"` passes the given option string *opts* to MetiTarski.

The script `addaxioms.pl` expands `include` directives at the source level in a set of problem files, creating a directory of problem files in which each occurrence of `include` has been replaced by the corresponding axioms. It can be useful for debugging, perhaps to eliminate axioms that you believe to be irrelevant in the hope of obtaining proof. The TPTP base directory is identified, as usual, by the `--tptp` option if it is provided and otherwise by the `TPTP` environment variable.

For example, one way to generate a directory of expanded problems is to visit the tptp directory and type the command

```
addaxioms.pl --in Problems --outdir TestDir --tptp .
```

You can add further axiom files to all problems by naming them in this command.

7 Input syntax and problem preparation

Problem syntax uses TPTP format extended with infix notation for arithmetic formulas. Decimal notation is accepted: a decimal such as `.23` abbreviates the fraction 23/100. To express a floating point number, write an expression such as `1.04e-18` or `4E5`. Parsing is done using ML-YACC. Please note some quirks of the grammar:

1. A `cnf` line introduces a clause, which *must* be enclosed within a pair of parentheses even if it consists of a single literal. Example:

   ```
   cnf(sqrt_squared, axiom, (X < 0 | sqrt(X)*sqrt(X) = X)).
   ```

2. The syntax for formulas only allows parentheses around a non-trivial formula. In particular, `(p(x))` and `((x=y))` are forbidden by the parser. But to allow for computer generated files in particular that have such redundant brackets, MetiTarski now has a preprocessor that specifically removes them. It is still better to avoid creating such files in the first place.

3. Free variables are forbidden in first-order formulas, that is, `fof` lines. All variables must be quantified.

7.1 Interval syntax

A special syntax for intervals is available.

```
t : (a,b)
```

This formula is logically equivalent to the conjunction $a < t \wedge t < b$. Here, t, a and b are all terms. Membership in a closed interval is written

```
t : (=a,b=)
```

The open and closed brackets can be mixed. Thus `X : (=0,1)` means $0 \leq X \wedge X < 1$.

7.2 The sine and cosine functions

The approximations for the sine and cosine functions become extremely inaccurate once their argument exceeds 6 in absolute value. Proofs concerning those functions outside of this range will almost certainly fail. For many applications, a proof can be obtained by generalising the problem to replace $\sin t$ and $\cos t$ by new variables X and Y subject to the constraint $X^2 + Y^2 = 1$. Note that a separate pair of variables is necessary for each different argument, t. The Chua problems are available both with and without this transformation.

We provide extended approximations for sin and cos as well, but the arguments must still be strictly limited. The comments in the axiom files give specific information concerning accuracy.

7.3 Miscellaneous remarks

The natural logarithm is written as ln. In contrast, the common logarithm (base 10) is written as log. It is defined in the axiom file `log.ax`. Note that many problem files with `log` in their name refer to the natural logarithm.

Unfortunately, the decision procedure is hyper-exponential in the number of variables, and MetiTarski is unlikely to be useful for problems containing more than five variables. A few nine-variable theorems have been proved.

MetiTarski can seldom prove equalities. Inequalities can only be exact at the point around which the power series of the relevant function has been expanded, typically 0 or 1.

8 Interpreting the output

By default, MetiTarski produces very little output, only a ticker indication of processor time used. You can suppress even this output, or produce traces in various levels of detail, using the `--verbose` option mentioned above. Detailed traces are not easy to interpret, even by experts.

When it terminates, MetiTarski prints a few statistics, including the processor time used. The proof is displayed if you requested this. There should also be a status line summarising the outcome. Naturally you want status Theorem, but other possibilities are Timeout and GaveUp, the latter meaning that MetiTarski ran out of clauses to process. Because MetiTarski is incomplete, we cannot conclude anything from its failure to prove a theorem. This can happen because you have not included necessary axioms, because the theorem is too difficult to prove, or because it is simply not amenable to the methods used by MetiTarski.

You can improve MetiTarski's performance by including only necessary axiom files (for example, don't include upper bounds if they aren't needed, which often is clear by inspection). Also, if you are trying to solve an engineering problem, writing decimal numbers to 10 significant figures will create extra work for MetiTarski to deliver an accuracy that can have no practical use. The necessary tolerances should be determined by your application,

and in most cases you should not need more than three significant figures. (Note that a standard resistor is only accurate to 10%.)

You will naturally want to try MetiTarski on your own problems, after converting them to MetiTarski syntax and inserting the necessary `include` directives. Use the supplied problems as examples. Please share any new problems, whether MetiTarski can solve them or not.

Acknowledgements

Other team members include Behzad Akbarpour, James Bridge and Grant Passmore. Research supported by the Engineering and Physical Sciences Research Council [grant numbers EP/C013409/1 and EP/I011005/1]. MetiTarski is a modified version of Joe Hurd's Metis prover.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] James Bridge and Lawrence Paulson. Case splitting in an automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 50(1):99–117, 2013.

[7] William Denman. QUANTUM: Qualitative abstractions of non-polynomial models. In Mehul Bhatt, Peter Struss, and Christian Freksa., editors, *27th International Workshop on Qualitative Reasoning*, pages 9–15, August 2013.

[8] 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.

[9] William Denman and César Muñoz. Automated real proving in PVS via MetiTarski. In Cliff Jones, Pekka Pihlajasaari, and Jun Sun, editors, *FM 2014: Formal Methods*, volume LNCS 8442, pages 194–199. Springer, 2014.

[10] William Denman, Mohamed H. Zaki, Sofiène Tahar, and Luis Rodrigues. Towards flight control verification using automated theorem proving. In Mihaela Bobaru, Klaus Havelund, GerardJ. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods*, volume LNCS 6617, pages 89–100. Springer, 2011.

[11] Rajeev Narayanan, Behzad Akbarpour, Mohamed H. Zaki, Sofiène Tahar, and Lawrence C. Paulson. Formal verification of analog circuits in the presence of noise and process variation. In *Design, Automation and Test in Europe, DATE 2010*, pages 1309–1312, 2010.

[12] 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.

[13] Grant Olney Passmore. *Combined Decision Procedures for Nonlinear Arithmetics, Real and Complex*. PhD thesis, University of Edinburgh, 2011.

[14] Lawrence C. Paulson. MetiTarski: Past and future. In Lennart Beringer and Amy P. Felty, editors, *ITP*, LNCS 7406, pages 1–10. Springer, 2012.

[15] Lawrence C. Paulson. Automated theorem proving for special functions: The next phase. In *Symposium on Symbolic-Numeric Computation*, SNC '14, pages 3–8, New York, NY, USA, 2014. ACM.