Verifying Hybrid Systems Involving Transcendental Functions

Paul Jackson[1], Andrew Sogokon[1], James Bridge[2], and Lawrence Paulson[2]

[1] School of Informatics, University of Edinburgh, UK
pbj@inf.ed.ac.uk, a.sogokon@sms.ed.ac.uk
[2] Computer Laboratory, University of Cambridge, UK
jpb65@cam.ac.uk, lp15@cam.ac.uk

Abstract. We explore uses of a link we have constructed between the KeYmaera hybrid systems theorem prover and the MetiTarski proof engine for problems involving special functions such as sin, cos, exp, etc. Transcendental functions arise in the specification of hybrid systems and often occur in the solutions of the differential equations that govern how the states of hybrid systems evolve over time. To date, formulas exchanged between KeYmaera and external tools have involved polynomials over the reals, but not transcendental functions, chiefly because of the lack of tools capable of proving such goals.

1 Introduction

KeYmaera is an interactive prover which makes use of external tools such as computer algebra systems for simplification, solving differential equations and proving quantified formulas involving real arithmetic. MetiTarski is a prover specifically tailored for reasoning with transcendental functions. It eliminates transcendental functions from inequalities by applying polynomial and continued-fraction bounds and employs external provers to discharge goals involving these approximations. In this section we will give an overview of the context which motivates the integration of these two systems.

1.1 Hybrid Systems

Hybrid systems generalise both transition systems and continuous dynamical systems. The state of a hybrid system has both discrete- and continuous-valued components. Together, the values of the discrete components specify the mode of the system. Within each mode the evolution of the state is governed by differential equations. Transitions between between modes usually have guards describing when they are enabled and specify how the continuous components might jump in value when the transitions are taken. Figure 1 shows an example hybrid system, described using the *hybrid automaton* formalism.

Hybrid systems are very useful for creating models of cyber-physical systems, systems which involve computers or some kind of discrete control logic interacting with a physical environment [7]. Cyber-physical systems are found in many

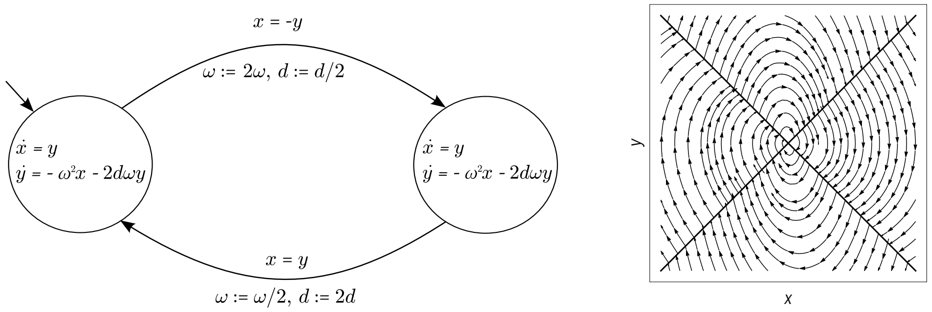
$x = -y$
$\omega := 2\omega,\ d := d/2$
$\dot{x} = y$
$\dot{y} = -\omega^2 x - 2d\omega y$
$\dot{x} = y$
$\dot{y} = -\omega^2 x - 2d\omega y$
$x = y$
$\omega := \omega/2,\ d := 2d$
y
x

Fig. 1. Switched damped oscillator (**left**) and a possible phase portrait (**right**).

industrial sectors, including transport, energy and health-care automation. They are frequently safety-critical, so there is much interest in improved verification techniques for them.

1.2 Formal Verification of Hybrid Systems

In the past two decades a variety of techniques have been explored for the formal verification of hybrid systems. Many have involved a bounded approach where one computes an over-approximation of the state space reachable after some number of interleaved time evolutions within modes and jumps between modes [8, 5, 15, 6]. The primary verification goal has been to show that no unsafe states are reached. Depending on the hybrid system considered, the chosen bound and the approximation methods, the state space exploration might reach a fixed-point, in which case verification of safety is sound. Otherwise, there's the possibility that there is an unsafe reachable state which has not yet been explored.

Since the early work there has been much improvement in the methods for representing and computing the over-approximations of the reachable state sets. Often these approaches have placed restrictions on the form of the differential constraints in modes and the jumps, requiring them to be linear or to be bounded by constants, for example. In these cases, systems with more general non-linear differential equations, perhaps involving transcendental functions, can be approximated with piecewise linear equations.

With KeYmaera [13] a different approach is investigated.

1.3 KeYmaera

KeYmaera mechanises a deductive calculus for reasoning about hybrid systems. The base calculus is *differential dynamic logic* ($\mathsf{d}\mathcal{L}$) [12]. It extends first-order logic with modalities $[\alpha]\phi$ and $\langle\alpha\rangle\phi$, where α is a *hybrid program* and ϕ is a $\mathsf{d}\mathcal{L}$ formula. Hybrid programs are described in a simple compositional language that includes conditional statements, loops, discrete state updates, and continuous state updates in which states evolve over a period of time according to differential equations. The modality $[\alpha]\phi$ asserts that ϕ holds after every run of α, the modality $\langle\alpha\rangle\phi$ asserts that ϕ holds after some run of α. The calculus augments the first-order logic rules with rules for handling the modalities and decomposing the structure of hybrid programs within the modalities.

The most common kinds of statements proved concern invariants of the systems. Proofs of such statements usually involve creation of inductive invariants and *differential invariants*. A differential invariant is a property of the system that can be established to hold over some interval of time by considering the truth of a certain auxiliary property at each point in time in the interval. Differential invariants are related to the concept of Lyapunov functions, generalised energy functions whose decrease in value over a state space region of interest is used to argue for stability in the theory of dynamical systems. Previous work in safety verification of hybrid systems introduced *barrier certificates* [14] which impose Lyapunov-like conditions on the time derivative of differentiable functions in order to prove safety properties. Differential invariants in turn generalise barrier certificates to formulas with boolean connectives [12] and thus allow one to work with a much larger class of invariants.

Proofs in KeYmaera can be guided interactively or can be automated using tableau-based strategies. KeYmaera includes heuristics for guessing simpler forms of inductive and differential invariants. The KeYmaera logic implements directly very little reasoning concerning expressions of real arithmetic and constraints on the derivatives of the real-valued state components. Instead, use is made of procedures in external tools such as the Mathematica computer algebra system and QEPCAD-B [2] for simplification of real expressions, solution of differential equations and proving goals involving real arithmetic.

To date the interface to these external tools has limited the expression language to real-valued polynomials, and has not permitted the use of transcendental functions such as sine, cosine, logarithm and exponentiation. This was primarily because there had been no effective techniques for proving goals that involved inequalities over expressions including transcendental functions. Such goals can arise in several ways. For example, transcendental functions can be used in the descriptions of hybrid systems. They are also commonly found in the solutions of linear differential equations. And there are examples of Lyapunov functions in the dynamical systems literature where transcendental functions are required.

The deductive approach taken in KeYmaera is harder to apply than the bounded automated approaches described in Section 1.2, as interaction and human-directed creative steps are needed. Its advantages include the possibility of proving richer properties, the lack of a restriction of analyses to some bound, and often better capabilities for exploring parameterised systems.

1.4 MetiTarski and Goals of Work

For several years, Paulson and others have been developing MetiTarski, an automatic proof engine specifically tailored for proving goals involving inequalities over transcendental functions [11, 10]. In the work reported here we are interested in exploring how MetiTarski could support reasoning about hybrid systems in KeYmaera. We are also hoping that transcendental problems generated from the hybrid systems domain can help steer the future development of MetiTarski.

Previously MetiTarski has been used for the verification of analog circuits, modeled as dynamical systems or hybrid systems [4]. The computer algebra system Maple was used to analyse continuous behaviours of the systems, to solve linearisations of the differential equations describing their time evolution, for example. A systematic partly-manual process was then used to set up the relevant goals for MetiTarski to solve. In the work reported here, KeYmaera provides a significantly richer, more automated framework for the top-level reasoning about hybrid systems and the coordination of external reasoning services.

The core of MetiTarski is a first-order resolution theorem prover Metis and a database of axioms specifying polynomial and rational function bounds on transcendental functions. Weights that guide the resolution are tailored so as to employ the axioms to reduce problems involving inequalities over transcendental functions to problems involving inequalities over real polynomial expressions. MetiTarski augments the resolution calculus with extra rules for handling real polynomial expressions. These rules make use of external tools for proving goals involving polynomial expressions, for example the Z3 SMT solver [9], QEPCAD-B [2], and the quantifier-elimination procedure provided by the Mathematica computer algebra system.

2 KeYmaera-MetiTarski interface

KeYmaera implements a plugin architecture (shown in Figure 2) in which the user may choose a backend tool to perform particular tasks, such as solving differential equations, simplifying arithmetic expressions and performing quantifier elimination.

The primary purpose of quantifier elimination is to prove goals involving quantified arithmetic expressions by reducing them to “true”. In KeYmaera, it is sometimes also useful to have quantifier elimination produce quantifier-free expressions involving variables that are free in the goals, as these quantifier-free expressions can suggest missing assumptions.

In our work we have added MetiTarski as a new quantifier elimination backend tool, handling the common case of when quantifier-elimination is expected to return “true”. The link is implemented in Scala and Java and uses a file-level interface in which first-order goals from KeYmaera are translated into MetiTarski's input format (a variation on the TPTP format that allows infix notation) and stored in temporary files. These files are passed as arguments to the MetiTarski binary along with command-line options which the KeYmaera user selects in KeYmaera's GUI. This link from KeYmaera to MetiTarski is now part of standard KeYmaera releases.

The diagram in Figure 2 labels with $\forall$ the interfaces where KeYmaera goals are universally closed and the quantifier-elimination procedure is only ever ex-

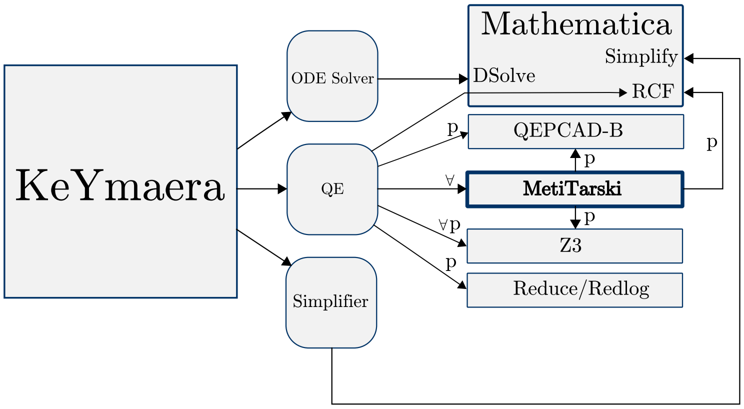

Fig. 2. KeYmaera plugin architecture

pected to return "true" when it succeeds. Label **p** is used for interfaces that handle purely polynomial problems, problems without special functions.

MetiTarski itself relies on decision procedures for real arithmetic and is able to call QEPCAD-B, Z3 and Mathematica to access this functionality. KeYmaera users may select the appropriate tool for MetiTarski by setting the pertinent option in KeYmaera. The problems sent to these external tools are purely polynomial, as shown in Figure 2.

In Figure 2, Mathematica is shown to provide more functionality than any of the other tools. In particular, it is able solve systems of differential equations. The user has to trust these solutions, but in certain cases this may introduce unsoundness.

The simplifier offered by Mathematica is very powerful and it may often be necessary to simplify complicated expressions before any further progress can be made on a problem using either Mathematica itself or MetiTarski. Once more, one needs to be aware of the potential soundness issues in performing this step.

While MetiTarski may use Mathematica's decision procedure for real closed fields (RCF), which it trusts to be sound, it will not make use of other potentially unsound computer algebra functionality, such as the simplifier.

3 Examples of how transcendental functions arise

We review here three ways in which transcendental functions can arise during formal verification of continuous and hybrid systems.

3.1 Systems with closed form solutions

Some systems admit closed-form solutions to the initial value problem; however, these tend to be much more complicated than the differential equations themselves and will often involve special functions.

KeYmaera offers inference rules which allow reasoning about safety and liveness properties by considering closed form solutions when they exist. Using this facility tends to generate first-order goals involving transcendental functions, which are delegated to an external solver.

We consider here a safety verification scenario where the solution is available in closed form and the safety property is ensuring boundedness of oscillation. The motion of a damped oscillator, such as that shown in Figure 3, can be described by the linear second-order differential equation

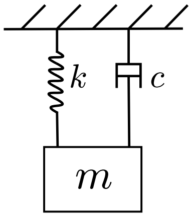

Fig. 3. Damped oscillator

$$\ddot{x} + 2d\omega\dot{x} + \omega^2 x = 0,$$

where $\omega = \sqrt{\frac{k}{m}}$ is the frequency, $d = \frac{c}{2\sqrt{km}}$ is the damping factor and x is the displacement from the point of equilibrium. We can convert this into a state space model by setting $x_1 = \dot{x}$ and $x_2 = x$. For a concrete example, let us choose $\omega = 2$ and $d = \frac{3}{5}$.

$$\begin{aligned}\dot{x}_1 &= -\frac{3}{5} \cdot 2 \cdot 2 \cdot x_1 - 2^2 \cdot x_2,\\ \dot{x}_2 &= x_1\end{aligned}$$

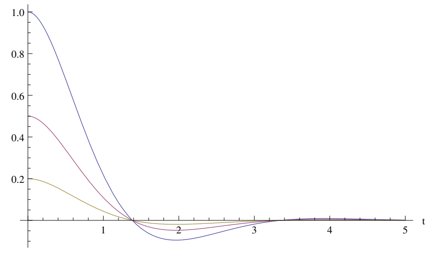

Fig. 4. x_2-component of solutions with $x_2(0) = \{1, \frac{1}{2}, \frac{1}{5}\}$, $x_1(0) = 0$.

It is intuitively obvious that a damped oscillator will lose energy and eventually come to a halt, assuming there is no input. Consider proving that an initial displacement x_2 will never result in that displacement subsequently being exceeded. We could phrase this property using differential dynamic logic as

$$t \geq 0,\ x_1 = 0,\ x_2 \leq b,\ x_2 \geq a \vdash [\dot{\boldsymbol{x}} = f(\boldsymbol{x})]\, x_2 \leq b.$$

Here the *box modality* $[\ ]$ expresses the property that $x_2 \leq b$ is necessarily true after the system evolves according to the system of differential equations $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ whenever it is initialised in a state satisfying the antecedent.

Consider the case where initial velocity is zero and the initial displacement is in the interval $[0, 1]$. A formalisation of this problem in KeYmaera is shown in Figure 5.

```
\programVariables{
   R x1;
   R x2;
}

\problem {
  (x2<=1 & x2>=0 & x1=0) ->
    \[ {x1' = -((3/5)*2*2*x1 + 2^2*x2), x2' = x1 } \]
     (x2<=1)
}
```

Fig. 5. Proving boundedness of displacement of a damped oscillator using KeYmaera.

Computing the solution, this amounts to proving

$$t \geq 0,\ x_2 \leq 1,\ x_2 \geq 0 \vdash \frac{1}{4}e^{-\frac{6}{5}t}x_2\left(4\cos(\frac{8}{5}t) + 3\sin(\frac{8}{5}t)\right) \leq 1.$$

This goal is difficult to prove, with Mathematica being unable to handle it in reasonable time; MetiTarski can solve this in under a second.

3.2 Transcendental functions in system description

In the previous example, we proved a property of a system by proving a property of the closed form solution to the differential equations governing evolution. It is not uncommon to encounter systems in which transcendental functions are used in the description of how system state continuously evolves. Transcendental functions can occur too in the description of the guards and state updates associated with mode switches in hybrid systems. Sometimes the descriptions of such systems can be transformed so as to eliminate the transcendental functions and have descriptions purely involving polynomial functions. In general though it is desirable to work directly with the transcendental functions.

It is rare that closed form solutions can be found for the continuous state evolution of systems described using transcendental functions. Indeed, it is also not possible to find closed form solutions for most systems described using nonlinear polynomials. To address these cases, a number of related methods have been developed that allow the proof of properties of interest by referring directly to the differential equations governing state evolution and not requiring solution of the equations. These methods use such concepts as *Lyapunov function*, *barrier certificate* and *differential invariant* (we refer the reader to the Appendix).

We give an example here of a simple dynamical system which involves transcendental functions in its description and sketch how an invariance property can be proven using differential invariants.

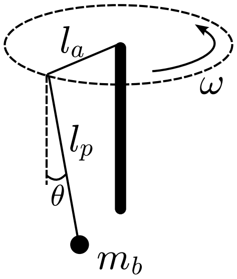

Fig. 6. Whirling pendulum

Consider a whirling pendulum (i.e. one which is itself suspended from a rod of radius l_a, moving with an angular velocity ω). Its equations of motion are given by the following non-polynomial system:

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= -\frac{k_f}{m_b}x_2 + \omega^2\sin(x_1)\cos(x_1) - \frac{g}{l_p}\sin(x_1),\end{aligned}$$

where the state x_1 is the pendulum's angle with the vertical and x_2 is the rate of change of this angle, k_f is the friction coefficient, l_p is the length of the rigid arm, and m_b is its mass (see [3] for a detailed description of the model). A possible Lyapunov function for this system suggested by Chesi [3] is

$$V(\boldsymbol{x}) = x_1^2 + x_1x_2 + 4x_2^2.$$

In KeYmaera we might formulate the property of time-evolution being confined to sub-level sets for a in the range $0\ldots b$ for some constant b using the sequent

$$0 \le a,\ a \le b,\ V(\boldsymbol{x}) \le a \ \vdash\ [\dot{\boldsymbol{x}} = f(\boldsymbol{x})]\ V(\boldsymbol{x}) \le a \quad .$$

This is most easily proved in KeYmaera by first rephrasing it as

$$0 \le a,\ V(\boldsymbol{x}) \le a \ \vdash\ [\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \wedge V(\boldsymbol{x}) \le b]\ V(\boldsymbol{x}) \le a \quad .$$

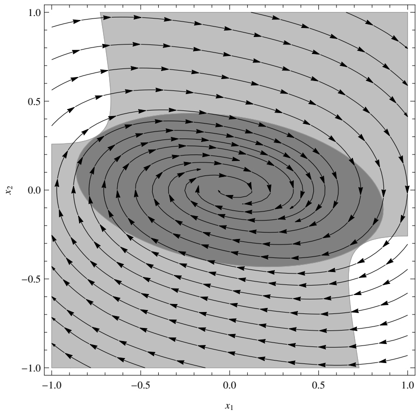

Fig. 7. Estimate to the domain of attraction $V(\boldsymbol{x}) \leq 0.69$ (dark shaded area) and states where $\nabla V \cdot f(\boldsymbol{x}) < 0$ (light shaded area).

```
\programVariables { R x1, x2, kf, mb, omega, g, lp, a; }

\problem {
   kf    = 0.2 & /* FRICTION */
   mb    = 1   & /* MASS OF RIGID ARM */
   omega = 0.9 & /* ROTATING ANGULAR VELOCITY */
   g     = 10  & /* GRAVITY ACCELERATION */
   lp    = 10  & /* LENGTH OF RIGID ARM */

 (x1^2 +x1*x2 +4*(x2^2)) <= a  & a>=0  /* LYAPUNOV FUNCTION */ ->

   \[ { x1' = x2,
        x2' = -(kf/mb)*x2 + (omega^2)*Sin(x1)*Cos(x1) - (g/lp)*Sin(x1) &
        (x1^2 +x1*x2 +4*(x2^2)) <= 0.69929971
      }
   \] (x1^2 +x1*x2 +4*(x2^2)) <= a
}
```

Fig. 8. Lyapunov function $V(\boldsymbol{x}) = x_1^2 + x_1 x_2 + 4x_2^2$ is non-increasing within the domain of attraction $V(\boldsymbol{x}) \leq 0.69929971$.

Taking $k_f = 0.2$, $m_b = 1$, $\omega = 0.9$, $l_p = 10$, gravity $g = 10$ and $b = 0.69929971$, we can formalise this property in KeYmaera as shown in Figure 8.

The number 0.69929971 defines the bound on the sub-level set of $V(\boldsymbol{x})$, which is used as a conservative estimate to the domain of attraction of the whirling pendulum.

Explicitly the subgoal we get in KeYmaera in applying the differential induction rule is

$$
\begin{aligned}
k_f = 0.2, m_b = 1,\ \omega = 0.9,\ g = 10,\ l_p = 10,\ x_1^2 + x_1x_2 + 4x_2^2 \le a,\ a \ge 0 \vdash \\
\forall x_{21}, x_{11} \in \mathbb{R}.\ x_{11}^2 + x_{11}x_{21} + 4x_{21}^2 \le 0.69929971 \implies \\
2x_{11}x_{21} + x_{21}^2 \\
+ x_{11}\big(- \frac{k_f}{m_b} x_{21} - \frac{g}{l_p} \sin(x_{11}) + \omega^2 \cos(x_{11}) \sin(x_{11})\big) \\
+ 8x_{21}\big(- \frac{k_f}{m_b} x_{21} - \frac{g}{l_p} \sin(x_{11}) + \omega^2 \cos(x_{11}) \sin(x_{11})\big) \ \le\ 0.
\end{aligned}
$$

MetiTarski solves this problem in under 10 minutes.

3.3 Non-polynomial invariant candidates

A further use case for the link between the two systems concerns the handling of invariant candidates which are non-polynomial.

Unlike in the previous example, where transcendental functions were used to define the dynamics of the system and the invariant candidate was polynomial, one may instead have a polynomial vector field and an invariant candidate featuring transcendental functions. An simple example is shown in Figure 9.

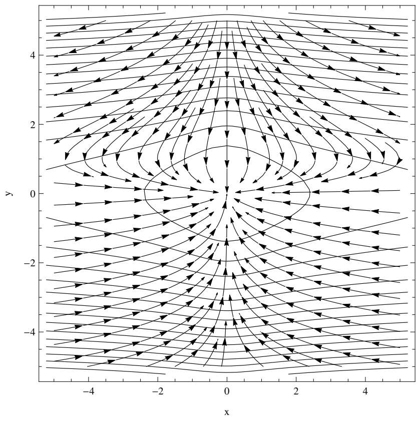

Fig. 9. Invariant sub-level sets of a non-polynomial Lyapunov function $V(x, y) = \ln(1 + x^2) + y^2$ [1].

Recently it has been shown that even for purely polynomial vector fields that are *globally asymptotically stable* it may be impossible to find a Lypaunov function which is of polynomial form [1].

Generally, allowing special functions in the description of invariant candidates enlarges the class of invariant assertions amenable to verification (given the right tools).

4 Performance and discussion

At present, MetiTarski and Mathematica are the only tools that are able to handle problems involving special functions which appear in KeYmaera proofs. Table 1 compares them on some of the problems featuring transcendental functions which arose during proof attempts in KeYmaera. The table shows the run-time in milliseconds of several tool configurations on the problems. Time here is wall-clock time on an Intel i5-2520M CPU @ 2.50GHz. A '-' character indicates that no result was obtained after running for 10 minutes. There are three columns for the MetiTarski results, each using a different external tool for proving polynomial problems. The name of the external tool is shown in parentheses in each case. The right-hand column shows how Mathematica performs when we pass it directly the problems with special functions.

Table 1. Problems involving transcendental functions in KeYmaera proofs.

Problem	Functions		MetiTarski		Mathematica
		(Z3)	(QEPCAD-B)	(Mathematica)	
Damped oscillator	$\exp, \sin, \cos$	430	850	2,403	-
Whirling pendulum	$\sin, \cos$	419,340	3,849	14,182	-
Domain of attraction	$\exp, \cos$	-	2,161	3,899	-
Drill string	$\sin, \cos$	17,441	30,270	48,944	-
Local Lyapunov	exp	-	-	-	-
Diffcut 1	exp	45	154	956	33
Diffcut 2	exp	-	-	-	59
Heater Simple	exp	144	376	1,427	68
Tunnel diode 1	exp	-	-	-	84,171
Tunnel diode 2	exp	227	370	1,587	18,444

MetiTarski handles well the first three problems featuring inequalities over trigonometric functions or a combination of trigonometric and exponential functions, whereas Mathematica times out on all these. The goal for the first problem, shown also earlier in Section 3.1, is

$$t \geq 0 \wedge x_2 \leq 1 \wedge x_2 \geq 0 \vdash \frac{1}{4}e^{-\frac{6}{5}t}x_2\left(4\cos(\frac{8}{5}t) + 3\sin(\frac{8}{5}t)\right) \leq 1.$$

MetiTarski proves this by using the bounding properties

$$\begin{aligned} 0 \le x \;\Rightarrow\; & \sin(x) \le x \\ & \cos(x) \le 1 - \tfrac{x^2}{2} + \tfrac{x^4}{24} \\ x \le 0 \;\Rightarrow\; & \quad e^x \quad \le \tfrac{2304}{(-x^3+6x^2-24x+48)^2}. \end{aligned}$$

MetiTarski's performance on the problems involving just the exponential function is more mixed. Consider the *diffcut 2* goal:

$$x > 15,\ t \ge 0 \;\vdash\; 25e^t + e^{t/2}(x-40) > 0.$$

MetiTarski's strategy of substituting polynomial or rational function bounds for exponential function occurrences is not so appropriate here, as the goal's validity depends on the relationship between e^t and $e^{t/2}$ for all $t \ge 0$. In general, polynomial or rational function bounds are accurate and best used for special function with bounded arguments, although we happened to have success with them in the *damped oscillator* example above where t is also unbounded. For *diffcut 2*, a simple solution strategy involves replacing $e^{t/2}$ with a new variable, and we are considering introducing such substitutions if we observe a significant number of further examples where they would useful.

The *tunnel diode 1* and *tunnel diode 2* goals are

$$\begin{aligned} t \ge 0 \;&\vdash\; -a_1e^{-k_1t} + a_2e^{-k_2t} - a_3e^{k_3t} \le 0, \\ t \ge 0 \;&\vdash\; -a_1e^{-k_1t} + a_2e^{-k_2t} + a_4e^{k_3t} \ge 0 \end{aligned}$$

respectively, where $a_1, a_2, a_3, a_4, k_1, k_2$ and k_3 are all positive constants. The constants a_3 and a_4 are both significantly larger than a_1 and a_2, and so the inequalities can be seen as obviously true from just basic bounding properties of the exponential function. MetiTarski has problems with the *tunnel diode 1* because the constants are not just rationals, but constant expressions involving square roots. For example a_1 is $1104311 - 34469\sqrt{254841}$. MetiTarski currently works with such constants by breaking them down and using bounding lemmas for the square root. MetiTarski has an interval constraint solver that can work with bounded interval approximations for constants, and it would easily handle such constants as above if we were to extend this facility to handle square roots.

The differences in MetiTarski's performance with different real polynomial arithmetic proof procedures appears primarily due to the differences in the procedures themselves and the interfaces to them. With virtually all the problems considered, the proof found by MetiTarski does not vary with the proof procedure selected. The lower performance of MetiTarski with Mathematica as the proof procedure for real arithmetic is due in part to the performance overhead incurred from contacting the Mathematica license server. KeYmaera keeps an open TCP connection with Mathematica, whereas MetiTarski needs to establish a new connection for each problem.

5 Conclusion

We have presented here some preliminary experiments with an interface between the KeYmaera and MetiTarski tools. The results are encouraging and we are now seeking more complex examples that bear a closer relationship to practical hybrid systems verification problems and that produce interesting problems for MetiTarski.

One issue is that often readily available examples have been reformulated or simplified so as to allow working only with polynomials. To this end we are finding we are having to develop expertise in how problems are first represented in KeYmaera and how KeYmaera is then guided to solving verification problems of interest. We are also trying to build closer ties with current users of KeYmaera. For example, we know of at least two groups applying KeYmaera to autonomous car problems that involve transcendental functions and we are hoping for fruitful collaboration with these groups.

Acknowledgements. This research was supported by EPSRC grants EP/I011005/1, EP/I010335/1. We would like to thank the anonymous reviewers for their feedback and helpful suggestions. We extend special thanks to Grant Passmore at the LFCS, University of Edinburgh, for offering his expert advice.

References

1. Ahmadi, A.A., Krstic, M., Parrilo, P.A.: A globally asymptotically stable polynomial vector field with no polynomial Lyapunov function. In: CDC-ECE. pp. 7579–7580 (2011)
2. Brown, C.W.: Qepcad b: a program for computing with semi-algebraic sets using cads. SIGSAM Bull. 37(4), 97–108 (Dec 2003), http://doi.acm.org/10.1145/968708.968710
3. Chesi, G.: Estimating the domain of attraction for non-polynomial systems via LMI optimizations. Automatica 45(6), 1536–1541 (2009)
4. Denman, W., Akbarpour, B., Tahar, S., Zaki, M., Paulson, L.: Formal verification of analog designs using metitarski. In: Formal Methods in Computer-Aided Design, 2009. FMCAD 2009. pp. 93–100 (2009)
5. Fränzle, M., Herde, C.: Hysat: An efficient proof engine for bounded model checking of hybrid systems. Formal Methods in System Design 30(3), 179–198 (2007)
6. Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: Spaceex: Scalable verification of hybrid systems. In: Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6806, pp. 379–395. Springer (2011)
7. Heemels, W., Lehmann, D., Lunze, J., De Schutter, B.: Introduction to hybrid systems. In: Lunze, J., Lamnabhi-Lagarrigue, F. (eds.) Handbook of Hybrid Systems Control – Theory, Tools, Applications, chap. 1, pp. 3–30. Cambridge University Press, Cambridge, UK (2009)
8. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: Hytech: A model checker for hybrid systems. STTT 1(1-2), 110–122 (1997)

9. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Tools and Algorithms for the Construction and Analysis of Systems, TACAS. LNCS, vol. 4963, pp. 337–340. Springer (2008)
10. Paulson, L.C.: MetiTarski: Past and Future. In: Beringer, L., Felty, A. (eds.) Interactive Theorem Proving, Lecture Notes in Computer Science, vol. 7406, pp. 1–10. Springer Berlin Heidelberg (2012)
11. Paulson, L.C.: http://www.cl.cam.ac.uk/ lp15/papers/Arith/. University of Cambridge (2013)
12. Platzer, A.: Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Springer, Heidelberg (2010)
13. Platzer, A.: http://symbolaris.com/info/KeYmaera.html. Carnegie Mellon Uniersity (2013)
14. Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: Alur, R., Pappas, G. (eds.) Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 2993, pp. 477–492. Springer Berlin Heidelberg (2004)
15. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation-based abstraction refinement. ACM Trans. Embedded Comput. Syst. 6(1) (2007)

Appendix: Direct methods and safety verification

Historically, Lyapunov was perhaps one of the first to observe that in the study of stability, closed form solutions are rarely revealing and that it is possible to work with the differential equation *directly* to prove properties of interest. This observation led to what has become known as Lyapunov's direct method, which introduced the concept of Lyapunov functions.

Informally, a Lyapunov function V is a continuously-differentiable *positive-definite* function of the system state, whose time-derivative along the vector field is never greater than zero. More precisely, given a system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ which is defined on some state space $X \subseteq \mathbb{R}^n$, if one can find a $V : X \to \mathbb{R}$ such that

$$\begin{aligned} V(\boldsymbol{x}) &> 0 \quad \forall \boldsymbol{x} \in X \setminus \mathbf{0}, \\ \nabla V \cdot f(\boldsymbol{x}) &\leq 0 \quad \forall \boldsymbol{x} \in X, \end{aligned}$$

then one can conclude that the origin is stable.

A set $\{\boldsymbol{x} \mid V(\boldsymbol{x}) \leq a\}$ is known as an a sub-level set of V and if V is a Lyapunov function, then each sub-level set of V is a system invariant (in forward time); that is, once a solution enters the set, it cannot escape.

The method of barrier certificates [14] uses Lyapunov-like conditions to argue for safety, rather than stability. Given a system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ as before, a set of initial states $X_i \subseteq X$ and a set of unsafe states $X_u \subseteq X$, if one can find a continuously-differentiable function $B : X \to \mathbb{R}$ such that

$$\begin{aligned} B(\boldsymbol{x}) &> 0 \quad \forall \boldsymbol{x} \in X_u, \\ B(\boldsymbol{x}) &\leq 0 \quad \forall \boldsymbol{x} \in X_i, \\ \nabla B \cdot f(\boldsymbol{x}) &\leq 0 \quad \forall \boldsymbol{x} \in X, \end{aligned}$$

then the system is guaranteed to be safe.

The problem of safety verification with barrier certificates is essentially that of finding a B which satisfies the above conditions.

The $d\mathcal{L}$ calculus used by KeYmaera provides a proof rule called *differential induction* (henceforth called DI; see Platzer [12] for a thorough exposition), which allows one to reason about invariance of sets defined by quantifier-free formulas,

$$\mathrm{DI}\frac{X \rightarrow \dot{F}}{F \rightarrow [\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \wedge X]F}.$$

In DI, F is a quantifier-free first-order formula in the theory of real arithmetic, X is the evolution domain constraint and the *differential formula* $\dot{F}$ is defined using the derivation operator D [12] which is given as follows:

$$\begin{aligned}
D(r) &= 0 \quad \text{for real numbers,}\\
D(x) &= \dot{x} \quad \text{for real variables,}\\
D(a+b) &= D(a) + D(b),\\
D(a \cdot b) &= D(a) \cdot b + a \cdot D(b),\\
D(F \wedge G) &\equiv D(F) \wedge D(G),\\
D(F \vee G) &\equiv D(F) \wedge D(G),\ (\wedge \text{ here is important for soundness})\\
D(a \leq b) &\equiv D(a) \leq D(b), \quad \text{accordingly for } \geq, >, <, = .
\end{aligned}$$

The differential formula $\dot{F}$ is shorthand for $D(F)^{f(\boldsymbol{x})}_{\dot{x}}$, where each $\dot{x}$ in $D(F)$ is replaced with the corresponding right hand side in the differential equation. Formulas F provable using DI are called *differential invariants*.

Safety verification with differential invariants is similar to the method of barrier certificates, i.e., given a formula F_{i} which is satisfied by the initial states and a formula F_{u} satisfied by the unsafe states, one requires a differential invariant F such that

$$\begin{aligned}
&F_{\mathrm{i}} \rightarrow F,\\
&F \rightarrow [\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \wedge X]F,\\
&F \rightarrow \neg F_{\mathrm{u}}.
\end{aligned}$$

Indeed, if one succeeds in finding a $F \equiv B(\boldsymbol{x}) \leq 0$, then this is equivalent to a proof of safety using barrier certificates. Differential invariants thus include barrier certificates as a special case [12].