Automated Theorem Proving For Special Functions: The Next Phase

Lawrence C. Paulson
Computer Laboratory
Univ. of Cambridge
United Kingdom
lp15@cl.cam.ac.uk

Categories and Subject Descriptors

[**Symbolic and algebraic manipulation**]: Symbolic and algebraic algorithms—*Theorem proving algorithms*; [**Theory of computation**]: Logic—*Logic and verification*

General Terms

theorem proving, verification, decision procedures

1. RESOLUTION THEOREM PROVING

Automated theorem proving, in a nutshell, is the combination of symbolic logic with syntactic algorithms. A formal proof calculus is chosen with two criteria in mind: expressiveness and ease of automation. These desiderata pull in opposite directions: Boolean logic and linear arithmetic are decidable, so the answers to all questions can simply be calculated, but these theories are not very expressive. At the other extreme, a dependent type theory such as the calculus of constructions used in Coq [6] is highly expressive and flexible, but complicates automation; even basic rewriting is difficult. Higher-order logic is often seen as a suitable compromise, expressive enough to reason directly about sets and functions, while still admitting substantial automation (especially in the case of Isabelle [18]).

Interactive theorem provers typically implement an expressive calculus, at least higher-order logic if not type theory, and rely on the user to guide the proof process. Automatic theorem provers, with few exceptions, are confined to first-order logic, for which highly refined and effective proof procedures exist. A venerable but still popular procedure is *resolution* [3]. The statement to be proved is negated and then transformed into a set (interpreted as a conjunction) of disjunctions. Finding the set of formulas to be inconsistent proves the desired theorem by contradiction.

For a simple example, suppose that we assume $P \vee Q$ and $P \rightarrow Q$ and seek to prove Q. After negating the conclusion, the problem amounts to the set $P \vee Q$, $\neg P \vee Q$, $\neg Q$. The resolution rule deduces new disjunctions by combining an atomic formula with its complement: from $P \vee Q$ and $\neg P \vee Q$ we deduce Q, which combined with $\neg Q$ yields the sought-after contradiction.

Resolution copes with first-order quantification easily. Consider the generalised problem that assumes

$$(\exists x\, P(x)) \vee (\exists x\, Q(x)) \quad \text{and} \quad \forall x\, [P(x) \rightarrow Q(x)]$$

and that seeks to prove $\exists x\, Q(x)$. The corresponding set of disjunctions is $P(a) \vee Q(b)$, $\neg P(x) \vee Q(x)$, $\neg Q(x)$, where some of the quantified variables have been replaced by constants and the remainder are implicitly universally quantified. Resolving the first two disjunctions yields $Q(a) \vee Q(b)$. From this, using $\neg Q(x)$, we deduce first $Q(b)$ and finally obtain a contradiction. We have *refuted* the negated conjecture, namely $\neg Q(x)$, thereby proving $\exists x\, Q(x)$, as desired.

Each resolution inference combines two disjunctions, say $P \vee A$ and $\neg Q \vee B$, where P and Q are atomic formulas that can be *unified*. This means finding a substitution σ such that $P\sigma$ and $Q\sigma$ are identical. Resolution then returns the disjunction $(A \vee B)\sigma$. Disjunctions in resolution are referred to as *clauses*. They are often written as sets of literals, where a *literal* is an atomic formula or its negation.

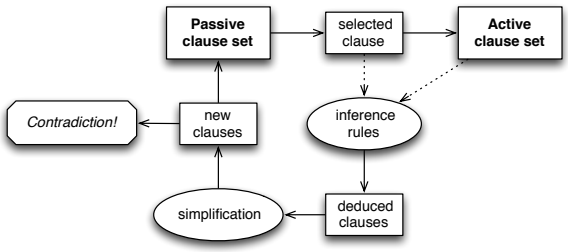

Figure 1: The main loop of resolution

A resolution prover's main loop (Fig. 1) manages two sets of clauses, *Active* and *Passive* [16]. The Active set consists of clauses that have been resolved with every other Active clause, while the Passive set consists of clauses waiting to be processed. At the start, all clauses belong to Passive. At each iteration, an element of the Passive set (called the *given* clause) is selected and moved to the Active set. The given clause is then resolved with every member of the Active set. Newly inferred clauses can be simplified, for example by rewriting, before they are added to the Passive set.

Innumerable refinements and sophisticated implementations have brought resolution to the point where it can prove theorems in many domains, in the presence of hundreds of

SNC'14, July 28–31, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2631948.2631950

axioms. Nevertheless, resolution theorem proving has not made a significant impact on mathematics. Its most celebrated achievement to date is settling the Robbins conjecture using EQP [15], a specialised theorem prover implementing a variant of resolution. The proof involved showing that three particular axioms were sufficient to characterise the notion of a Boolean algebra. This sort of highly syntactic, self-contained problem is typical of what can be achieved using fully automatic methods. Although the Robbins conjecture was of real significance, questions about the sufficiency of particular sets of axioms are a niche concern. They lie well outside of mainstream mathematics.

2. METITARSKI

MetiTarski [1] is a resolution theorem prover specialised to the domain of the real numbers. It proves first-order formulas built up over real inequalities that may involve the familiar special functions such as sin, cos, $\tan^{-1}$ and ln. Its strategy is to replace occurrences of special functions by upper or lower bounds for them, given by axioms. These bounds are typically truncated Taylor series or rational functions derived from continued fraction approximations. Eliminating occurrences of division leaves us with polynomial inequalities, which in many cases can be decided by calling an external decision procedure for real-closed fields (RCF), such as QEPCAD [9].

All of this takes place within the resolution framework. The original problem is broken down into clauses. The selection of suitable upper or lower bound axioms happens automatically, matching inequalities in the axioms with those in the problems. A collection of bounds, valid over a variety of intervals, linear or of high degree, is typically given to MetiTarski, and resolution alone determines which bounds are used in the final proof. But MetiTarski also has special code to convert arithmetic expressions to a canonical form (in an ad hoc manner), to isolate function occurrences so that they can easily be eliminated, and to flatten expressions involving division to assist in its elimination. The actual replacement of division by multiplication is performed by resolution, given the obvious axioms relating the two operations.

In general, a resolution step creates a new clause that has more literals than the two existing clauses. Many resolutions are possible, but very few of them are useful. In order to focus the search, Bachmair and Ganzinger [2] introduced ordering heuristics for restricting each resolution step to particular literals. Resolution therefore eliminates those literals first. In the specific case of MetiTarski, we arrange things to eliminate literals containing special functions, replacing them by their upper or lower bounds as appropriate. Once division has also been eliminated from a literal, it becomes a polynomial inequality. This is the stage at which QEPCAD, or another external decision procedure, can be called. The point is to identify literals that must be false (within their context) and can therefore be safely deleted. This creates a fine-grained integration between resolution and computer algebra.

The *absolute value* function is specified by the obvious pair of clauses:

$$x < 0 \vee |x| = x \qquad 0 \leq x \vee |x| = -x$$

Using these, MetiTarski replaces $|t|$ by t or $-t$. The effect is to perform case analysis on the sign of t, but it happens naturally according to the standard mechanisms are resolution theorem prover. Given a clause of the form

$$P(|t|) \vee C,$$

two new clauses can be generated:

$$P(t) \vee t < 0 \vee C$$
$$P(-t) \vee 0 \leq t \vee C$$

An instance of the absolute value function has been eliminated. In particular, if C and $P(t)$ can both be refuted, then the first clause will be reduced to just $t < 0$. Refuting $P(-t)$ will reduce the second clause to $0 \leq t$, which will contradict $t < 0$ and conclude the proof. Note that MetiTarski identifies $t < u$ with $\neg(u \leq t)$.

Finally, MetiTarski augments resolution with backtracking, as seen in SAT and SMT solvers [8]. Case splitting effectively solves subproblems that naturally arise during the search. It is appropriate because many MetiTarski clauses have no universally quantified variables. Thus, MetiTarski has an unusual architecture combining elements of resolution and SAT-solving, tightly integrated with external decision procedures for real closed fields.

3. A WORKED EXAMPLE

The best way to see how MetiTarski actually works is by examining a small example:

$$\forall x\, |e^x - 1| \leq e^{|x|} - 1$$

The first step is to negate the conjecture, which in effect asserts the existence of a counterexample, c. Our objective is to refute the following inequality:

$$e^{|c|} < 1 + |e^c - 1| \tag{1}$$

According to the absolute value mechanism described above, a clause is generated corresponding to the case $c < 0$:

$$0 \leq c \vee e^{-c} < 1 + |e^c - 1|$$

(you may prefer to read it as $c < 0 \implies e^{-c} < 1 + |e^c - 1|$.) Note that $|c|$ has been replaced by $-c$.

A second use of absolute value reasoning generates a clause corresponding to the case $e^c - 1 < 0$:

$$1 \leq e^c \vee 0 \leq c \vee e^{-c} < 2 - e^c$$

(Note that MetiTarski transforms arithmetic expressions in a fairly obvious manner.)

Now two occurrences of the exponential function have been isolated and can be eliminated using the upper or lower bound axioms. Among these axioms is a well-known inequality, $1+x \leq e^x$. So in particular, $1-c \leq e^{-c}$. The MetiTarski axioms are formulated to include one step of transitivity reasoning. Thus, resolving $1 - c \leq e^{-c}$ with the previous clause yields (after simplification)

$$1 \leq e^c \vee 0 \leq c \vee e^c < 1 + c.$$

The same lower bound can be used again, this time in the form $1 + c \leq e^c$. Transitivity yields $1 + c < 1 + c$, which of course is false, so the resulting clause is simply

$$1 \leq e^c \vee 0 \leq c.$$

We have eliminated an exponential. We have made progress!

It should be clear that the two literals in $1 \leq e^c \vee 0 \leq c$ are equivalent and that one of them ought to be deleted.

MetiTarski accomplishes this in a very unintuitive manner, using the upper bound

$$e^x \leq 2304/(-x^3 + 6x^2 - 24x + 48)^2,$$

which is asserted to be valid for $x \leq 0$. (It is actually valid for $x \leq 3.192\ldots$, but MetiTarski strengthens the condition for pragmatic reasons.) Transitivity now yields

$$1 \leq 2304/(-c^3 + 6c^2 - 24c + 48)^2 \vee 0 < c \vee 0 \leq c.$$

Now the division operator must be eliminated. Division axioms exist for various combinations of inequalities and signs. Assuming that the divisor is positive (the opposite assumption leads to a dead end) yields the following clause:

$$(-c^3 + 6c^2 - 24c + 48)^2 \leq 2304 \\ \vee (-c^3 + 6c^2 - 24c + 48)^2 \leq 0 \vee 0 < c \vee 0 \leq c.$$

The next step is critical. MetiTarski examines the first, complicated, literal in the context of its disjunction with the other literals, which means, under the assumption that $c < 0$. The external decision procedure considers the formula

$$\exists x \left[x < 0 \wedge (-x^3 + 6x^2 - 24x + 48)^2) \leq 2304\right]$$

and finds it to be false.[1] Therefore the first literal is inconsistent with its context and can be deleted. A similar step deletes the literals $(-c^3 + 6c^2 - 24c + 48)^2 \leq 0$ and $0 < c$. MetiTarski has concluded that the supposed counterexample is nonnegative:

$$0 \leq c.$$

We are not finished. Proof search in resolution works in strange ways, and here it goes right back to the conjecture clause (1) and generates a clause corresponding to the case $e^c - 1 \geq 0$:

$$e^{|c|} < e^c \vee e^c < 1$$

A further use of absolute value reasoning, now for the case $c \geq 0$, yields $c < 0 \vee e^c < e^c \vee e^c < 1$, which immediately simplifies to

$$c < 0 \vee e^c < 1.$$

A further use of $1 + x \leq e^x$ produces $c < 0 \vee 1 + c < 1$ which simplifies to $c < 0$. But this contradicts $0 \leq c$ and we are finished. QED.

This mechanical process will never produce an elegant proof or solve any problem requiring variable transformations or geometric insights. Nevertheless, it is sufficient to prove an enormous number of complicated problems. Note that the proof sketched above has been greatly simplified for purposes of presentation, and that MetiTarski may generate many different proofs depending on various settings.

4. APPLICATIONS

It should be clear from the proof presented above that MetiTarski is most likely to be applied to engineering problems, where elegant proofs are neither required nor expected to exist. In an engineering design, safety constraints will require certain tolerances to be observed; any special function inequalities that arise in such a proof will probably hold for mundane reasons.

[1]The negations of the other literals should be included in the conjunction. Redundant here, they are omitted for clarity.

A serious obstacle to finding engineering applications, however, is the enormous computational cost of the real-closed field decision procedure. Because it is inherently doubly exponential in the number of variables [12], problems having more than four or five variables are typically infeasible. And the variables include not just those that describe the system's dynamics, but its parameters.

Nevertheless, MetiTarski is starting to find applications in verification problems, for example in air traffic control [14]. The idea here is to automate the Nichols plot technique for proving that a control system is stable. More recent work [13] involves the integration of MetiTarski with PVS [19], an interactive theorem prover that is heavily used to verify aerospace software, particularly at NASA. MetiTarski turns out to be very much faster than the automation already available on PVS, but the integration between the two systems currently requires the result of MetiTarski to be trusted [13]. This is undesirable in applications that require the highest degree of confidence, and the authors propose using MetiTarski for what might be called the prototyping of a proof. For final runs, slower alternatives [17] that generate fully-checked PVS proofs can be substituted.

This suggests that a potential role for MetiTarski is in combination with interactive theorem proving, where problems that are too difficult to tackle in a single step can be manually broken down into pieces that MetiTarski can handle, and save considerable user effort compared with the difficulty of finding a fully manual proof. Given the correctness concerns raised above, the first thing we will want to do is satisfy ourselves that MetiTarski's axioms are true. Proving them within a particular interactive theorem prover is the first step towards a trustworthy integration of MetiTarski into that prover.

5. UPPER AND LOWER BOUNDS

MetiTarski requires upper and/or lower bounds for all real-valued functions appearing in the problem [1]. In some cases, such as the sine and cosine functions, these can be obtained from Taylor series. Under fairly general conditions, the Taylor expansion of a function is an alternating series where the truncations alternate between upper and lower bounds. For the exponential, logarithm and inverse tangent functions, MetiTarski uses continued fraction approximants [11]. These are much more accurate than the Taylor series approximants, and are valid over wider ranges. But the underlying theory of continued fractions (or the closely related Padé approximants [5]) is complicated.

A simple way to prove that an approximation is an upper (or lower) bound of a function is to form the difference and then differentiate. If the difference is zero at a given point and elsewhere the derivative is always positive (or negative), then the desired conclusions are easily obtained. But it is not obvious (at least to me) how to differentiate a continued fraction, obtaining a closed form for an arbitrary approximant. The corresponding task for a power series, of course, is trivial.

A simple workaround for such issues is to note that MetiTarski uses a fixed, finite collection of bounds. It does not dynamically generate increasingly accurate approximants in the course of the proof, but draws from axiom files that have been prepared, one could even say curated, for this purpose. We therefore have only a finite number of bounds to examine. In the early days, I am sorry to say, visual inspection

using a computer algebra system was the only proof available for some of these bounds. Now, the necessary inequalities have been mechanically proved (using Isabelle) for the most important functions: sin, cos, $\tan^{-1}$, ln, exp and square root.

The exponential function is perhaps the most interesting case. It is obvious that no polynomial or rational function upper bound for e^x can hold for all $x \geq 0$. We content ourselves with bounds that are valid for finite intervals. MetiTarski has a selection of such upper bounds, each of which is a rational function where the denominator eventually vanishes. The most complicated of these is valid for $0 \leq x < 9.94\ldots$. (Upper bounds valid for all $x < 0$ pose no difficulty, of course.)

Let us consider a simple case, the third continued fraction approximant (Fig. 2):

$$\mathrm{cf3}\, x \triangleq -\frac{x^3 + 12x^2 + 60x + 120}{x^3 - 12x^2 + 60x - 120}$$

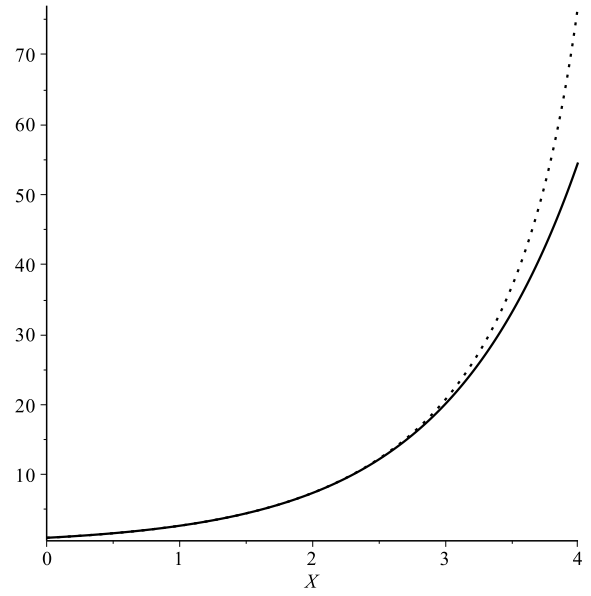

Figure 2: A continued fraction upper bound for e^x

As mentioned above, a simple way to prove that this is an upper bound is to form the difference and differentiate. When the function in question is the logarithm or inverse tangent, then differentiation eliminates it ($\frac{d}{dx} \ln x = 1/x$; $\frac{d}{dx} \tan^{-1} x = 1/(1+x^2)$). The resulting polynomial inequality can be solved with little effort. The exponential function is not so easy to eliminate because $\frac{d}{dx} e^x = e^x$. A simple trick is to take the logarithm of both sides, since applying a monotonic function preserves an inequality. The exponential disappears, replaced by a logarithm elsewhere. The derivative of this difference is very well-behaved:

$$\frac{d}{dx}\big[\ln(\mathrm{cf3}\, x) - x\big] = -\frac{x^6}{(x^3 - 12x^2 + 60x - 120)(x^3 + 12x^2 + 60x + 120)}$$

The derivative is positive provided $x^3 - 12x^2 + 60x - 120 < 0$ and in particular if $0 < x < 4.64\ldots$, when we can conclude that $e^x < \mathrm{cf3}\, x$.

Similar methods suffice to prove that $\mathrm{cf3}\, x \leq e^x$ for $x \leq 0$. There are a few complications. The derivative has another singularity at $x = -4.64\ldots$, and for smaller values of x we even have $\mathrm{cf3}\, x \leq 0$, so the logarithm is undefined. Nevertheless, this proof is also straightforward. Using case analysis on the sign of $x^3 + 12x^2 + 60x + 120$ (which is positive provided $x > -4.64\ldots$), the positive case can be tackled by taking logarithms as above. In the opposite case, $\mathrm{cf3}\, x \leq e^x$ is trivial. Note that this case analysis does not involve an explicit expression for the single real root of $x^3 + 12x^2 + 60x + 120$.

6. CORRECTNESS CONCERNS

Users of various sorts of mathematical software have varying standards of correctness. At one extreme, floating-point arithmetic is susceptible to rounding errors, and responsibility for getting meaningful answers is entirely devolved to programmers. Computer algebra systems can be seen in the same light: their operation is opaque, involving interactions among a great many algorithms, and it is the user's responsibility to check that results make sense.

At the other extreme, interactive theorem provers are typically constructed to have a minimal inference kernel that has to be trusted. The rest of the source code operates through the kernel, and by construction, cannot introduce logical reasoning errors. This strict architecture is complemented by methodological constraints: users are strongly encouraged to work exclusively with definitions and never to assume axioms. It is not unusual to hear people claim that following such a programme delivers infallible results. (Therefore, unfortunately, it is also not unusual to see unrealistic models formalised and incorrectly stated theorems proved.) Resolution theorem provers and SMT solvers have no minimal influence kernel, but any reasoning error is still regarded as a serious fault.

MetiTarski compounds these difficulties. As a theorem prover, correctness is of the utmost importance. And yet, it depends on computer algebra algorithms (the external decision procedures) to solve polynomial subproblems. The axioms of upper and lower bounds were generated using the computer algebra system Maple, given definitions of continued fraction expansions published online [4]. So there are many potential sources of errors. On the other hand, MetiTarski generates highly detailed proofs in which all logical reasoning is shown explicitly. The references to axioms and external decision procedures are all made explicit, along with MetiTarski's internal arithmetic simplification steps.

The previous section has addressed the question of the correctness of the axioms. Many of these inequalities are familiar to the cognoscenti, but creating formal proofs of each axiom (in the form that is given to MetiTarski) is necessary to achieve full confidence here. Verifying the axioms using Isabelle/HOL has opened up the possibility of integrating these two reasoning tools.

The chief remaining difficulty is how to verify the results given by the external RCF decision procedures. MetiTarski is chiefly concerned with RCF problems that involve only existential quantifiers[2] and we are only interested in proving such formulas to be inconsistent. (Equivalently, we wish to prove universally quantified formulas.) Therefore, the decision procedure delivers a result without any supporting ev-

[2]An experimental extension to mixed-quantifier problems is seldom used.

idence. An analogy may be useful: consider the question of whether some very large number N is prime or not. We call an external procedure, and if it returns a supposed factor k then we can easily check this claim, but if it declares N to be prime then we are none the wiser.

A key research question, then, is how to instrument the external decision procedure so that its workings can be verified. They work by producing a *cylindrical algebraic decomposition*: a partitioning of the space into regions, in each of which the formula has a constant value, and then checking the formula at a sample point in each region. Computing this decomposition is computationally expensive; could it be returned to be checked by an independent tool? But at present, it is not clear how a supposed decomposition can be confirmed to be correct without repeat the entire computation again.

This might be seen as an instance of a more general question, how can we verify negative results delivered by computer algebra algorithms?

7. THE NEXT PHASE

As mentioned above, an integration between MetiTarski and PVS has turned out to be useful [13]. Its main drawback is the need to accept MetiTarski proofs on faith. And we have considered the question of how to prove MetiTarski's axioms using elementary means, along with the more vexing question of how to verify claims made by the external RCF decision procedures. The verification of the axioms is now largely complete, although it gives rise to additional questions. Approximation theory has naturally focused on getting the most accurate approximations, but other criteria turn out to be important:

- approximations that are upper and lower bounds of a given function
- coverage of a variety of intervals
- a choice between highly accurate approximations and less accurate but simpler ones

Given that MetiTarski has already been integrated with PVS, it is likely that some future work will continue to involve PVS. But I am an Isabelle developer and my work will inevitably focus on that system, in which the necessary instances of the continued fraction expansions have already been verified. The idea of formalising some of the general theory of continued fractions and Padé approximants is appealing, even if it requires formalising substantial chunks of complex analysis and similar material. Formalised libraries of such core mathematical material are well overdue.

The question of RCF decision procedures has already seen some attention [7, 10], mainly in the context of the Coq proof assistant [6]. Decision methods for real-closed fields have been implemented within Coq itself, but they are not efficient enough for practical use. Different approaches are needed, and what shape they will take is currently unclear.

In unpublished work, Grant Passmore has been developing a self-contained RCF decision procedure within MetiTarski itself. It justifies its results within a specialised formal calculus. These proofs are designed with the objective of being easy to check within a system such as Isabelle, while minimising the amount of mathematics that must be formalised. Currently, it handles only the univariate case, but if this step is successful then the bivariate case could be tackled, and so forth.

The goal of integrating MetiTarski with Isabelle motivates a general programme of research that includes reasonably effective methods for deciding polynomial inequalities. That will ultimately require the formalisation of a great deal of real algebraic geometry. A small amount of computer algebra machinery already exists in a number of interactive theorem provers, and it greatly facilitates this sort of work. We can predict a bootstrapping process whereby the formalisation of core mathematics and the sound implementation of core computer algebra functions proceed in tandem.

Acknowledgements. The Edinburgh members of the team are Paul Jackson, Grant Passmore and Andrew Sogokon. The Cambridge team includes James Bridge, William Denman and Zongyan Huang. We are grateful to our outside collaborators, including César Muñoz, Eva Navarro-López and André Platzer. Research supported by the Engineering and Physical Sciences Research Council [grant numbers EP/I011005/1, EP/I010335/1].

8. REFERENCES

[1] 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.

[2] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[3] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, 2001.

[4] F. Backeljauw, S. Becuwe, M. Colman, A. Cuyt, and T. Docx. Special functions: continued fraction and series representations, 2008. On the Internet at `http://www.cfhblive.ua.ac.be/`.

[5] George A. Baker, Jr. *Essentials of Padé Approximants*. Academic Press, 1975.

[6] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Springer, 2004.

[7] Yves Bertot, Frédérique Guilhot, and Assia Mahboubi. A formal study of Bernstein coefficients and polynomials. *Mathematical Structures in Computer Science*, 21(04):731–761, 2011.

[8] 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.

[9] Christopher W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *SIGSAM Bulletin*, 37(4):97–108, 2003.

[10] Cyril Cohen and Assia Mahboubi. Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination. *Logical Methods in Computer Science*, 8(1), 2012.

[11] A. Cuyt, V. Petersen, B. Verdonk, H. Waadeland, and W.B. Jones. *Handbook of Continued Fractions for*

Special Functions. Springer, 2008.

[12] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. Symbolic Comp.*, 5:29–35, 1988.

[13] 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.

[14] 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.

[15] W. McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[16] William McCune and Larry Wos. Otter: The CADE-13 competition incarnations. *Journal of Automated Reasoning*, 18(2):211–220, 1997.

[17] César Muñoz and Anthony Narkawicz. Formalization of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, 51(2):151–196, 2013.

[18] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002. LNCS Tutorial 2283.

[19] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. PVS: Combining specification, proof checking, and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *Computer Aided Verification: 8th International Conference, CAV '96*, LNCS 1102, pages 411–414. Springer, 1996.