UNIVERSITY OF BATH

Citation for published version:
Davenport, J, Bradford, R, England, M & Wilson, D 2012, Program Verification in the presence of complex numbers, functions with branch cuts etc. in *Proceedings of SYNASC 2012: 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing.* IEEE, Piscataway, pp. 83-88, SYNASC 2012: 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 25/09/12. https://doi.org/10.1109/SYNASC.2012.68

DOI:
10.1109/SYNASC.2012.68

Publication date:
2012

Document Version
Peer reviewed version

Link to publication

© 2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

University of Bath

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 24. Sep. 2020

Program Verification in the presence of complex numbers, functions with branch cuts etc.

James H. Davenport, Russell Bradford, Matthew England & David Wilson
Department of Computer Science
University of Bath
Bath, BA2 7AY, U.K.
Email: {J.H.Davenport,R.J.Bradford,M.England,D.J.Wilson}@bath.ac.uk

***Abstract*—In considering the reliability of numerical programs, it is normal to "limit our study to the semantics dealing with numerical precision" (Martel, 2005). On the other hand, there is a great deal of work on the reliability of programs that essentially ignores the numerics. The thesis of this paper is that there is a class of problems that fall between these two, which could be described as "does the low-level arithmetic implement the high-level mathematics". Many of these problems arise because mathematics, particularly the mathematics of the complex numbers, is more difficult than expected: for example the complex function log is not continuous, writing down a program to compute an inverse function is more complicated than just solving an equation, and many algebraic simplification rules are not universally valid.**

The good news is that these problems are *theoretically* capable of being solved, and are *practically* close to being solved, but not yet solved, in several real-world examples. However, there is still a long way to go before implementations match the theoretical possibilities.

***Keywords*-Verification; algebra; branch cut; singularity;**

I. Introduction

It is customary, even though not often explicitly stated, to think that programming errors in numerical programs come in three distinct flavours, which we can categorise as follows.

1. **Blunder** (of the coding variety) This is the sort of error traditionally addressed in "program verification": are all array elements properly initialised before use, "fence post" errors[1], are array bounds always respected etc.? These problems are essentially independent of the numerics of the problem, and indeed are normally taught/described in integer contexts.
2. **Parallelism** Issues of deadlocks or races occurring due to the parallelisation of an otherwise correct sequential program. Again, these problems are essentially independent of the numerics of the problem.
3. **Numerical** Do truncation and round-off errors, individually or combined, mean that the program computes approximations to the "true" answers which are out of tolerance. This is the area traditionally addressed in Numerical Analysis. There are really two subquestions here: the rounding question, i.e. does $\mathbf{R}_{IEEE}$ (or whatever other arithmetic we are using) approximate $\mathbf{R}$ sufficiently well, and the truncation error question, e.g. is the discretisation h small enough that it is the mathematical ϵ or is $\sum_1^N$ equivalent to $\sum_1^\infty$. Unfortunately the two interact; for example reducing h in $f'(x) \approx \frac{f(x+h)-f(x)}{h}$ to reduce the truncation error increases the rounding problems.

We note that [1], taken as a specimen of the verification literature, contains 30 papers, of which only [2] deals with strictly numerical issues, four with parallelism issues, and the rest (83%) with the first kind.

It is the thesis of this paper that there is a fourth kind of error, which we can describe as follows

4. **Manipulation** A piece of algebra, which is "obviously correct", turns out not to be correct when interpreted, not as abstract algebra, but as the manipulation of functions $\mathbf{R} \to \mathbf{R}$ or $\mathbf{C} \to \mathbf{C}$.

Note: throughout this paper we take the standard definitions of the branch cuts of the elementary functions from [3], [4] (as tightened in [5]). Other definitions would have different, but not fewer, problems. We also use the Anglo-Saxon convention that $\log$ etc. (and $\sqrt{}$) denote single-valued functions ($\log 1 = 0$, $\sqrt{4} = 2$), whereas Log etc. denote multi-valued functions ($\mathrm{Log}(1) = \{2k\pi i : k \in \mathbf{Z}\}$, $\mathrm{Sqrt}(4) = \{2, -2\}$).

The problems we are going to describe arise largely (though not entirely[2]) from complex numbers, and it is sometimes said "real programs don't use complex numbers", despite the fact that the introduction of `COMPLEX` into Fortran II was probably the first instance of a language data type that did not correspond to a machine data type. The authors know of several major uses of complex numbers and complex analysis, in particular many problems which

[1] From the old puzzle "A farmer wishes to make a 100-metre fence with supporting posts every 10 metres — how many posts are needed", to which the answer is 11, since each end needs a post.

[2] See section IV for a counter-example.

arise in fluid dynamics, where two-dimensional real space $\mathbf{R}^2 = \{(x, y)\}$ is viewed as the complex plane $\mathbf{C} = \{z = x + iy\}$. It is then normal to map this copy of $\mathbf{C}$ to another (in which the variable is traditionally denoted w or ζ) where the problem is easier to solve. Such an analytic map $z \mapsto w$ is termed a *conformal* map.

II. The Kahan Example

This example comes from [6, pp. 187–189], and the ultimate motivation is fluid flow in a slotted strip (z space), which we wish to transform to a more convenient shape (w space).

With the usual definitions, the necessary conformal map

$$w = g(z) := 2\,\mathrm{arccosh}\left(1 + \frac{2z}{3}\right) - \mathrm{arccosh}\left(\frac{5z+12}{3(z+4)}\right) \tag{1}$$

is only the same as the ostensibly more efficient

$$w \stackrel{?}{=} q(z) := 2\,\mathrm{arccosh}\left(2(z+3)\sqrt{\frac{z+3}{27(z+4)}}\right), \tag{2}$$

if we avoid the teardrop shaped area

$$\left\{ z = x+iy : |y| \le \sqrt{\frac{-(x+3)^2(2x+9)}{2x+5}}, -\frac{9}{2} \le x \le -3 \right\} \tag{3}$$

as shown by Figure 1

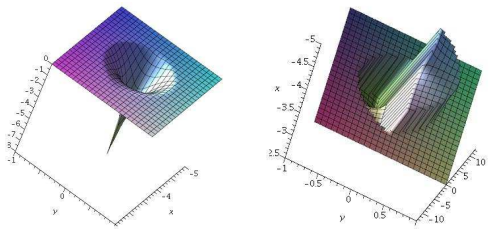

Figure 1. Plots of the real and imaginary parts of $g(z) - q(z)$.

In fact, most computer algebra systems will refuse, these days, to convert one into the other, but this does not constitute a proof of difference.

Challenge 1: Demonstrate automatically that g and q are not equal, by producing a z at which they give different results.

The technology described in [7] would answer this question by decomposing the complex plane into regions, via means of cylindrical algebraic decomposition (CAD) with respect to the branch cuts of the functions, and testing the identity at a sample point in each region

A fully-automated implementation for this example has yet to be produced since the geometry can be quite involved. (See section II for details.) However, progress is currently being made on the problem. The `BranchCuts` package [8] developed at Bath and to be included with Maple 17 does isolate the curve

$$y = \pm\sqrt{\frac{(x+3)^2(-2x-9)}{2x+5}} \tag{4}$$

with the appropriate x range as a potential obstacle (it is the branch cut of q). However, it is just one of a set of cuts returned by the code. The plotting option in the package produces Figure 2 which presents the teardrop and the entire real axis as potential cuts.

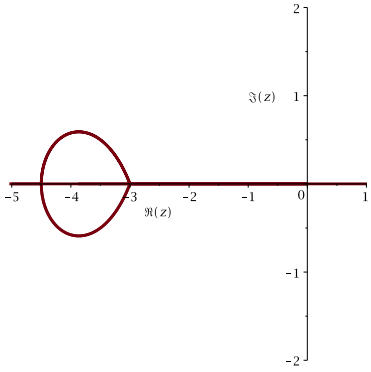

Figure 2. Plot of the potential branch cuts for $g(z) = q(z)$ produced by the `BranchCuts` package.

The package calculates cuts for functions by mapping the defining branch cuts of a function by the argument. The output is generated using Maple's `solve` facilities, and the user can choose to look for solutions in the complex variable, or to first split into real and imaginary parts and work over the reals. The first method is quicker and more widely applicable, but the second produces more intuitive output including the algebraic description of the teardrop in equation (4).

The package identifies the potential branch cuts of a composition of functions (a sum, a product or an equality) as the union of the cuts for individual components. Hence the identification of the real axis as a potential obstacle is not surprising since the individual terms do have branch cuts here, with the resulting discontinuities happening to cancel out in the composition.

However, the output described here would not be suitable for the technology of [7] since the input into CAD must be semi-algebraic. We can modify the code to just return the polynomial equalities and inequalities that define each set of cuts. For this example, there are 7 such sets, one of which consists of the three relations below.

$$4y(2y^2x + 2x^3 + 5y^2 + 21x^2 + 72x + 81) = 0$$

$$
\begin{aligned}
&4(y^4 - x^4 + 3y^2x - 13x^3 \\
&\quad +9y^2 - 63x^2 - 135x - 108) < 0 \\
&4x^4 - 4y^4 - 12y^2x + 52x^3 \\
&\quad -63y^2 + 225x^2 + 324x < 0
\end{aligned} \tag{5}
$$

Figure 3 gives a plot of these three curves. By testing sample points we see where the inequalities are satisfied and infer that the branch cut defined is the teardrop along with the real axis above -3. These issues and the implementation of the `BranchCuts` package are discussed further in [8].

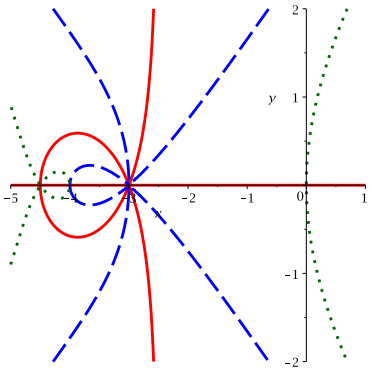

Figure 3. Plot of the information in (5). The solid line is the equality, the dashed line the first inequality and the dotted line the second.

If we pass the full set of polynomials to CAD (ignoring whether they are equalities or inequalities) then clearly a lot more information will be processed than required. Using the variable ordering $y > x$ and the command `CylindricalAlgebraicDecompose` within Maple 16 [9], this produces a CAD of 409 cells. Given (3) we might hope for a minimal CAD of 13 cells, or if we accept that the real axis must be included in any calculations then a minimal CAD of 19 cells, (see Figure 4).

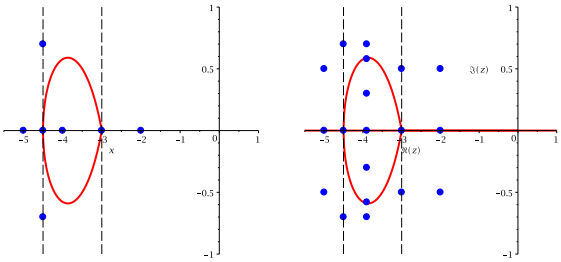

Figure 4. Possible minimal CADs for Kahan's example. The dots indicate sample points for a cell.

Note that there is a simplification of g *valid over the whole complex plane* : g can legitimately be rewritten to

$$
w = h(z) := 2 \ln \left(\frac{1}{3} \frac{\sqrt{3\,z+12}\left(\sqrt{z+3}+\sqrt{z}\right)^2}{2\sqrt{z+3}+\sqrt{z}} \right), \tag{6}
$$

The technology in [7] can show this, i.e. $\forall z \in \mathbf{C}\, g(z) = h(z)$, but there are multiple square roots requiring denesting [26, §4.3] and (as formulated) square roots in the denominator. Indeed the standard tools of Maple 16 currently get this wrong: `coulditbe(g<>h);` returns `true`, which *ought* to indicate that there is a counter-example.

Challenge 2: Demonstrate automatically that g and h are equal.

Although the technology in [7], implemented in a mixture of Maple and QEPCAD (though we are working on a purely Maple implementation based on [9]), may be able to meet this challenge in time, we would be left with the problem of trusting the underlying demonstration code. So there is the additional problem of translating this methodology into a tool such as MetiTarski [10].

III. Joukowski Examples

Consider the Joukowski map [11, pp. 294–298]:

$$
f : z \mapsto \frac{1}{2}\left(z + \frac{1}{z}\right). \tag{7}
$$

A. Joukowski (a)

Lemma 1: f is injective as a function from $D := \{z : |z| > 1\}$.

If $z \mapsto \zeta$ then $1/z \mapsto \zeta$, and there are no other pre-images of ζ (since the algebraic inverse of (7) is the solution of a quadratic). If $|z| > 1$, then $|1/z| < 1$, so z is unique in D.

In fact f is a bijection from D to $\mathbf{C}^\ddagger := \mathbf{C} \setminus [-1, 1]$, and hence has an inverse.

Of course, (7) is the conformal map $\mathbf{C} \to \mathbf{C}$ that equates to the map

$$
f_R : (x, y) \mapsto \left(\frac{1}{2}\,x + \frac{1}{2}\,\frac{x}{x^2+y^2}, \frac{1}{2}\,y - \frac{1}{2}\,\frac{y}{x^2+y^2} \right) \tag{8}
$$

$\mathbf{R}^2 \to \mathbf{R}^2$. However, it is not obvious from (8) alone that f_R is a bijection $D \to \mathbf{C}^\ddagger$, i.e. that

$$
\begin{aligned}
\forall x_1 \forall x_2 \forall y_1 \forall y_2 \quad & \Big(x_1^2 + y_1^2 > 1 \wedge x_2^2 + y_2^2 > 1 \wedge \\
& x_1 + \tfrac{x_1}{x_1^2+y_1^2} = x_2 + \tfrac{x_2}{x_2^2+y_2^2} \wedge \\
& y_1 - \tfrac{y_1}{x_1^2+y_1^2} = y_2 - \tfrac{y_2}{x_2^2+y_2^2} \Big) \Rightarrow \\
& \big(x_1 = x_2 \wedge y_1 = y_2 \big).
\end{aligned} \tag{9}
$$

Challenge 3: Demonstrate automatically the truth of (9), which is also [12, (49)].

We have been unable to do this with either the QEPCAD [13] implementation of Partial Cylindrical Algebraic Decomposition [14] or the Maple implementation of Cylindrical Algebraic Decomposition via triangular decomposition [9].

Brown [15] suggested writing problems of the form $\forall \mathbf{v} A \to (f = 0 \wedge g = 0)$ as $\neg\exists \mathbf{v} A \wedge \neg(f = 0 \wedge g = 0)$. This splits into two clauses: [12, §7.2], which can be seen to have equational constraints [16].

Figure 5. Maple's `solve` on inverting Joukowski

```
> [solve(zeta = 1/2*(z+1/z), z)];
```

$$\left[\zeta + \sqrt{\zeta^2-1}, \zeta - \sqrt{\zeta^2-1}\right]$$

Figure 6. Maple's actual `solve` on inverting injective Joukowski

```
> [solve(zeta = 1/2*(z+1/z), z)]\
   assuming abs(z) > 1
```

$$\left[\zeta + \sqrt{\zeta^2-1}, \zeta - \sqrt{\zeta^2-1}\right]$$

Even these are beyond QEPCAD and Maple currently. However, Brown [15] has been able to recast these clauses (manually) to make them amenable to QEPCAD, and indeed solved the problem in under 12 seconds.

Challenge 4: Automate these techniques and transforms. Having established (or not) that f is a bijection $D \to \mathbf{C}^\ddagger$, we want its inverse. Formally, this is trivial, as one referee said

> The inverse of Joukowski is the solution of a quadratic with the usual sign ambiguity:

if $\zeta = \frac{1}{2}\left(z + \frac{1}{z}\right)$, then $2z\zeta = z^2 + 1$ and $z = \zeta \pm \sqrt{\zeta^2 - 1}$. This is easily within the grasp of computer algebra, as seen in Figure 5. The only challenge might be the choice implicit in the $\pm$ symbol: which do we choose?

Unfortunately, the answer is "neither", or at least "neither, uniformly". The true answer is that, for f a bijection from $\{z : |z| > 1\}$ to $\mathbf{C} \setminus [-1, 1]$, its inverse is

$$f_1(\zeta) = \zeta \begin{cases} +\sqrt{\zeta^2-1} & \Im(\zeta) > 0 \\ -\sqrt{\zeta^2-1} & \Im\zeta) < 0 \\ +\sqrt{\zeta^2-1} & \Im(\zeta) = 0 \wedge \Re(\zeta) > 1 \\ -\sqrt{\zeta^2-1} & \Im(\zeta) = 0 \wedge \Re(\zeta) < -1 \end{cases} \tag{10}$$

In fact, a better (at least, free of case distinctions) definition is

$$f_2(\zeta) = \zeta + \sqrt{\zeta - 1}\sqrt{\zeta + 1}. \tag{11}$$

The techniques of [7] are able to **verify** (11), in the sense of showing that $f_2(f(z)) - z$ is the zero function on $\{z : |z| > 1\}$.

Challenge 5: Derive automatically, and demonstrate the validity of, either (10) or (11). In terms of Maple, we would want to see Figure 7, rather than the actual Figure 6.

Figure 7. Ideal Maple `solve` on inverting injective Joukowski

```
> solve(zeta = 1/2*(z+1/z), z)\
   assuming abs(z) > 1
```

$$\zeta + \sqrt{\zeta - 1}\sqrt{\zeta + 1}$$

In terms of derivation, the techniques of [17] seem worthy of investigation, but the first author has been unable to do this derivation satisfactorily by this route.

B. Joukowski (b)

Here the function is again given by (7).

Lemma 2: f is injective as a function from $H := \{z : \Im z > 0\}$.

As in Lemma 1, if $z \mapsto \zeta$ then $1/z \mapsto \zeta$, and there are no other pre-images of ζ. If $\Im(z) > 0$, $\Im(1/z) < 0$, and f in therefore injective from H.

In fact, f is a bijection from H to $\mathbf{C}\backslash((-\infty, -1]\cup[1, \infty))$, and hence has an inverse.

Again, it is not obvious from (8) alone that f_R is a bijection, now from $\{(x, y)|y > 0\}$, i.e. that

$$\begin{aligned} \forall x_1 \forall x_2 \forall y_1 \forall y_2 \qquad & \Big(y_1 > 0 \wedge y_2 > 0 \wedge \\ & x_1 + \tfrac{x_1}{x_1^2+y_1^2} = x_2 + \tfrac{x_2}{x_2^2+y_2^2} \wedge \\ & y_1 - \tfrac{y_1}{x_1^2+y_1^2} = y_2 - \tfrac{y_2}{x_2^2+y_2^2} \Big) \\ & \Rightarrow \big(x_1 = x_2 \wedge y_1 = y_2\big). \end{aligned} \tag{12}$$

Challenge 6: Demonstrate automatically the truth of (12). It is likely that the ideas of [15] can do this, but again these need automation.

We have the same challenge over the inverse of f: again formally it is $f^{-1} \stackrel{?}{=} \zeta \pm \sqrt{\zeta^2 - 1}$, and the only challenge is the $\pm$ symbol: which do we choose? Here [11, (5.1-13), p. 298] argues for

$$f_3(\zeta) = \zeta + \underbrace{\sqrt{\zeta - 1}}_{\arg\in(-\pi/2,\pi/2]} \underbrace{\sqrt{\zeta + 1}}_{\arg\in(0,\pi]}. \tag{13}$$

Challenge 7: Find a way to represent functions such as $\underbrace{\sqrt{\zeta+1}}_{\arg\in(0,\pi]}$

Fortunately this one is soluble in this case[3], we can write $\underbrace{\sqrt{\zeta+1}}_{\arg\in(0,\pi]} = i \underbrace{\sqrt{-\zeta-1}}_{\arg\in(-\pi/2,\pi/2]}$, and the latter is the normal `sqrt` function of [3]. Hence we have an inverse function

$$f_4(\zeta) = \zeta + \sqrt{\zeta - 1} i \sqrt{-\zeta - 1}. \tag{14}$$

Challenge 8: Demonstrate automatically that this is an inverse to f on $\{z : \Im z > 0\}$.

There is also the problem of deducing (14). One could try automatic deduction on the lines of [17], but there is another possibility: heuristic search followed by verification *if* the verification were feasible [18].

[3]And is probably soluble more generally, but the authors know of no general work on "alternative formulations".

IV. A Real Example

Just in case the reader thinks that the real numbers are immune from these problems, consider the addition rule for the inverse tangent, quoted as ([3, (4.4.34)] [4, (4.24.15)])

$$\mathrm{Arctan}(x) \pm \mathrm{Arctan}(y) = \mathrm{Arctan}\left(\frac{x \pm y}{1 \mp xy}\right).$$

Despite the caveat in [4] that "The above equations are interpreted in the sense that every value of the left-hand side is a value of the right-hand side and vice versa", it is in fact the case that the 'obvious' two equations are true separately, `viz.`

$$\mathrm{Arctan}(x) + \mathrm{Arctan}(y) = \mathrm{Arctan}\left(\frac{x+y}{1-xy}\right) \quad (15)$$

$$\mathrm{Arctan}(x) - \mathrm{Arctan}(y) = \mathrm{Arctan}\left(\frac{x-y}{1+xy}\right) \quad (16)$$

Consider (15): This is valid for the multi-valued Arctan, but for the single-valued arctan only when $|1 - xy| < 1$, due to a "branch cut at infinity" of arctan. Nevertheless, the single-valued version of (15) is often cited as true: see for example [19, (5.2.5)].

Over the reals, this is a non-challenge, the techniques of [7] do solve it easily, and produce a counterexample.

V. So why are these challenges?

A. Complex functions and branch cuts

These are difficult subjects, which have tended to be brushed under the carpet. The first truly algorithmic approach is ten years old ([20], refined in [7]), and has various difficulties.

1) At its core is the use of Cylindrical Algebraic Decomposition of $\mathbf{R}^N$ to find the connected components of $\mathbf{C}^{N/2} \setminus \{\text{branch cuts}\}$. The complexity of this is doubly exponential in N: upper bound of $d^{O(2^N)}$ [21] and lower bounds of $2^{2^{(N-1)/3}}$ [22], [23]. While better algorithms for the connected components problem are in principle known ([24] is $d^{O(N\sqrt{N})}$), we do not know of any accessible implementations.
   Furthermore, we are clearly limited to small values of N, at which point looking at $O(\ldots)$ complexity is of limited use. We note that the cross-over point between $2^{(N-1)/3}$ and $N\sqrt{N}$ is at $N = 21$. A more detailed comparison is given in [21]. Hence there is a need for practical research on low-N Cylindrical Algebraic Decomposition.
2) While the fundamental branch cut of $\log$ is simple enough, being $\{z = x + iy | y = 0 \wedge x < 0\}$, actual branch cuts are messier. Part of the branch cut of (2) is

$$2x^3 + 21x^2 + 72x + 2xy^2 + 5y^2 + 81 = 0 \wedge$$
$$\text{other conditions},$$

whose solution accounts for the curious expression in (3). While there has been some progress in manipulating such images of half-lines (described in [25], [26]), there is almost certainly more to be done.

B. Injectivity

Lemmas 1 and 2 might seem to be statements about complex functions of one variable, so why do we need to handle (or fail to handle) statements about four real variables to prove them? There are three, rather distinct, reasons for this.

1) The statements require the $|\cdot|$ function (Lemma 1) or the $\Im$ function (Lemma 2), neither of which are complex analytic functions. Hence some recourse to real analysis (and therefore twice as many variables) seems inevitable, though it would be nice to have a more formal statement and proof of this.
2) Equations (9) and (12) are the direct translations of the basic definition of injectivity. In practice, certainly if we were looking at functions $\mathbf{R} \to \mathbf{R}$, we would want to use the fact that the function concerned was continuous.
   Challenge 9: Find a better formulation of injectivity questions $\mathbf{R}^N \to \mathbf{R}^N$, making use of the properties of the functions concerned (certainly continuity, possibly rationality).
3) While equations (9) and (12) are statements from the existential theory of the reals, and so the theoretically more efficient algorithms quoted in [21] are in principle applicable, the more modern developments described in [27] do not seem to be directly applicable. However, we can transform then into a disjunction of statements to each of which the Weak Positivstellensatz [27, Theorem 1] is applicable.
   Challenge 10: Solve these problems using the techniques of [27],

VI. Conclusions

The aim of this paper has been to demonstrate that translating mathematical problems into programs may require some algebraic manipulations whose accuracy is not as obvious as one might think, and whose verification is *currently* not as straightforward as we would like, despite the fact that their correctness is, in principle, decidable. A summary is given in Table I.

These are, largely, concrete challenges that, we hope, will spur practical advances in this domain.

Acknowledgement

The authors would like to thank Acyr Locatelli, Gregory Sankaran and Nicolai Vorobjov of the Bath Triangular Sets seminar, as well as Scott McCallum (Macquarie U.) who kindly visited us, for their input, Chris Brown (U.S. Naval Academy) for [15], and the referees for their comments, but the errors and omissions are all the authors.

Table I
CURRENT STATE OF THESE CHALLENGES

Challenge	State
1	Mathematically solved by [7], geometry taxes current solvers.
2	Mathematically solved by [7], branch cut determination not yet implemented.
3/6	Mathematically solved by [28, etc.], geometry defeats current solvers.
4	Under development.
5/8	Mathematically solved [7], geometry defeats current solvers, and is probably significantly harder than in the previous problems.
7	Unknown: probably straightforward research project.
9	Unknown: research project.
10	Unknown: project for the authors of [27].

REFERENCES

[1] R. Cousot (Ed.), "Verification, Model Checking, and Abstract Interpretation," *Springer Lecture Notes in Computer Science 3385*, 2005.

[2] M. Martel, "An Overview of Semantics for the Validation of Numerical Programs," in *Proceedings Verification Model Checking and Abstract Interpretation. Springer Lecture Notes in Computer Science 3385*, 2005, pp. 59–77.

[3] M. Abramowitz and I. Stegun, "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing," *US Government Printing Office*, 1964.

[4] National Institute for Standards and Technology, "The NIST Digital Library of Mathematical Functions," http://dlmf.nist.gov, 2010.

[5] R. Corless, J. Davenport, D. Jeffrey, and S. Watt, "According to Abramowitz and Stegun, or arccoth needn't be uncouth," *SIGSAM Bulletin 2*, vol. 34, pp. 58–65, 2000.

[6] W. Kahan, "Branch Cuts for Complex Elementary Functions," in *Proceedings The State of Art in Numerical Analysis*, A. Iserles and M. Powell, Eds., 1987, pp. 165–211.

[7] J. Beaumont, R. Bradford, J. Davenport, and N. Phisanbut, "Testing Elementary Function Identities Using CAD," *AAECC*, vol. 18, pp. 513–543, 2007.

[8] M. England, R. Bradford, J. Davenport and D. Wilson, "Understanding branch cuts of expressions," *in preparation*, 2013, http://opus.bath.ac.uk/32511/.

[9] C. Chen, M. Moreno Maza, B. Xia, and L. Yang, "Computing Cylindrical Algebraic Decomposition via Triangular Decomposition," in *Proceedings ISSAC 2009*, J. May, Ed., 2009, pp. 95–102.

[10] L. Paulson, "MetiTarski: Past and Future," in *Proceedings Interactive Theorem Proving*, 2012, pp. 1–10.

[11] P. Henrici, "Applied and Computational Complex Analysis I," *Wiley*, 1974.

[12] D. Wilson, "Polynomial System Example Bank," http://opus.bath.ac.uk/29503, 2012.

[13] C. Brown, "QEPCAD B: a program for computing with semi-algebraic sets using CADs," *ACM SIGSAM Bulletin 4*, vol. 37, pp. 97–108, 2003.

[14] G. Collins and H. Hong, "Partial Cylindrical Algebraic Decomposition for Quantifier Elimination," *J. Symbolic Comp.*, vol. 12, pp. 299–328, 1991.

[15] C. Brown, "Re: Query about QEPCAD," *Personal Commnication to David Wilson*, 2012.

[16] S. McCallum, "On Projection in CAD-Based Quantifier Elimination with Equational Constraints," in *Proceedings ISSAC '99*, S. Dooley, Ed., 1999, pp. 145–149.

[17] R. Corless and D. Jeffrey, "The Unwinding Number," *SIGSAM Bulletin 2*, vol. 30, pp. 28–35, 1996.

[18] S. Gulwani, "Why not use Heuristics?" Question at SYNASC 2012, 2012.

[19] D. Terr, "Math is Amazingly Powerful," http://www.mathamazement.com/Lessons/Pre-Calculus/05_Analytic-Trigonometry/sum-and-difference-formulas.html, 2012.

[20] R. Bradford, R. Corless, J. Davenport, D. Jeffrey, and S. Watt, "Reasoning about the Elementary Functions of Complex Analysis," *Annals of Mathematics and Artificial Intelligence*, vol. 36, pp. 303–318, 2002.

[21] H. Hong, "Comparison of several decision algorithms for the existential theory of the reals," Tech. Rep. 91-41, 1991.

[22] C. Brown and J. Davenport, "The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition," in *Proceedings ISSAC 2007*, C. Brown, Ed., 2007, pp. 54–60.

[23] J. Davenport and J. Heintz, "Real Quantifier Elimination is Doubly Exponential," *J. Symbolic Comp.*, vol. 5, pp. 29–35, 1988.

[24] S. Basu, M.-F. Roy, M. Safey El Din, and E. Schost, "A baby step-giant step roadmap algorithm for general algebraic sets," http://arxiv.org/abs/1201.6439, 2012.

[25] N. Phisanbut, R. Bradford, and J. Davenport, "Geometry of Branch Cuts," *Communications in Computer Algebra*, vol. 44, pp. 132–135, 2010.

[26] N. Phisanbut, "Practical Simplification of Elementary Functions using Cylindrical Algebraic Decomposition," Ph.D. dissertation, University of Bath, 2011.

[27] G. Passmore and P. Jackson, "Combined Decision Techniques for the Existential Theory of the Reals," in *Proceedings Intelligent Computer Mathematics*, J. C. *et al.*, Ed., 2009, pp. 122–137.

[28] G. Collins, "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition," in *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, 1975, pp. 134–183.