Adherence is Better than Adjacency: Computing the Riemann Index Using CAD

James C. Beaumont, Russell J. Bradford, James H. Davenport & Nalina Phisanbut [*]
Dept. of Computer Science, University of Bath
Bath BA2 7AY England
{J.Beaumont, R.J.Bradford, J.H.Davenport, cspnp}@bath.ac.uk

ABSTRACT

Given an elementary function with algebraic branch cuts, we show how to decide which sheet of the associated Riemann surface we are on at any given point. We do this by establishing a correspondence between the Cylindrical Algebraic Decomposition (CAD) of the complex plane defined by the branch cuts and a finite subset of sheets of the Riemann surface. The key advantage is that we no longer have to deal with the difficult 'constant problem'.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms

General Terms

Algorithms, Theory

Keywords

Riemann Surfaces, Elementary Functions, Branch Cuts

1. INTRODUCTION

The elementary functions are the field of functions obtained by applications of exp, log and the arithmetic operations to a set of variables $x_1, \ldots, x_n$ and constants, $\mathbb{C}$. As in previous papers by the authors[4, 8], we shall focus particularly on those elementary functions which are in fact multi-valued. We shall use the notation that terms such as log and $\sqrt[n]{}$, and more generally, f, h, denote single-valued functions from $\mathbb{C}$ to $\mathbb{C}$, whilst Log, ${}_n$Sqrt and F, H denote multi-valued functions, regarded as mapping $\mathbb{C}$ into sets of values, so that $\text{Sqrt}(z) = \{w : w^2 = z\} = \{\pm\sqrt{z}\}$ for example. Numerous well-known identities for multi-valued functions exist; examples include $\text{Log}(z^2) - \text{Log}(z) - \text{Log}(z) = \{0\}$, and $\text{Sqrt}(z^2) = \{\pm z\}$. The $=$ is of course to be interpreted as set equality. As pointed out in [15] not all such formulae are identities and instead require set inclusions: we have that $\text{Log}(z^2) \supset 2\,\text{Log}(z)$ for example. Many others may be found in [1], which we will use to provide a set of realistic test formulae— exercising occasional care, however, in their interpretation[16]. Given $H \subseteq 0$, the problem is then to decide on what regions of $\mathbb{C}^n$ does $h = 0$ hold? The paper [15] provides a graphic illustration that one is indeed forced to consider the geometry of $\mathbb{C}^n$ with respect to the branch cuts of h in order to answer this question.

One important application of having a method to decide such questions is to the area of *simplification*. To obtain and define precisely what one means by a 'simplification', is an old and difficult problem[23]. We shall not grapple with this issue here but refer instead to recent progress in [9] for a potential approach, and to [8] for some of the problems involved with working with multi-valued transformations.

1.1 Previous Work

Progress towards constructing a verification system for multi-valued formulae as described above has been reported in [4] and its precursors. This is based on the Decomposition Method first suggested in [19], which requires one to:

1. calculate the set of branch cuts of the proposed identity $F = 0$;
2. find a sample point in each of the regions in $\mathbb{C}$ defined by the cuts;
3. evaluate the identity numerically using that point, thereby concluding whether the formula is true or not on that entire region by the Monodromy theorem.

Further details of how each of the steps above should be performed can be found in [4]. A key point to remember is that, as first suggested in [8], we use Cylindrical Algebraic Decomposition (CAD; see [10]) for step two; we shall assume that the reader is familiar with the basic notions involved in this algorithm. To do this, we restrict the class of formulae under consideration to those where the branch cuts are algebraic: it is sufficient to prohibit anything other than n^{th} roots from being nested inside other elementary functions.[1]

It is however worth pointing out that step one has now been made computationally efficient by the recent proposal to use resultants to eliminate n^{th} roots in the input expression; see [5] for details. Also in that paper, the use of an efficient method[18] to decide which is the best projection

[*]The authors gratefully acknowledge the support of EPSRC, under grant number GR/R84139/01.

[1]Note that this applies to constant functions; witness $\log(x - \exp(2))$, which has a non-algebraic branch cut.

order is demonstrated to be of great importance regarding the efficiency of step two. Thus in this paper, we are still strongly advocating a CAD based Decomposition Method; what we are proposing here is a new and more efficient approach to performing the final step. This step is surprisingly non-trivial as was seen in [3, 4] and so an improvement is highly desirable. We first provide a convenient summary of the most serious problems involved at this stage.

Suppose for simplicity of exposition that our formula H is of one $\mathbb{C}$ variable. Then we can compute a description of the branch cuts of h, as a semi-algebraic set in 2 real variables. Firstly suppose that the region we are testing has co-dimension > 0. In the terminology of CAD, this means that we are investigating a region that comprises a particular section, s say, of a stack. Let the sample point of s be $p = (x, y)$, which must be interpreted as the complex number $x + iy$. As argued in [8], numerical evaluation of $H(p)$ may give completely incorrect results except in the rare cases where x, y have finite floating point representations. A symbolic approach would therefore seem necessary, but in the worst case scenario, the point p will have some coordinates that cannot be expressed in terms of radicals. Any attempt to search for alternative sample points in sections where this is possible— those constructed over level 1 regions of full dimension— would result in an undesirable coupling between steps 2 and 3 of the algorithm. Even in the cases where p is expressible using radicals one runs into the 'constant problem' and has to generally resort to algorithms that are potentially costly and rely on the truth of number-theoretic conjectures[26]. The constant problem is in fact undecidable for sufficiently large function fields[25]. This is inhibiting as ultimately, one would like to handle non-elementary multi-valued functions as well, such as the W-Lambert function[14] for example. Secondly, in all regions, it may happen that p is an 'unlucky sample point'. If H is a rational function of elementary functions H_i each of which has the set of branches $\{h_i\}_i$ then p is an unlucky point if for one or more of the h_i we have that $h_i(p)$ are equal for several different branches. If we use such a p, then we cannot guarantee that we can draw correct conclusions about the truth of the identity H on the region it represents. An example is afforded by $p(z)\operatorname{Log}(q(z)^2) - 2p(z)\operatorname{Log}(q(z))$ where $p, q \in \mathbb{C}[z]$ and the unlucky points are then the roots of p, q. The method of [27], as was demonstrated in [3], is an effective, although costly, solution to this problem. Finally the work reported in [5] demonstrated that the number of cells that are produced by problems of seemingly simple appearance can be extremely large. This of course exacerbates the problems mentioned above. In such cases, numerical evaluation can also be very time-consuming— assuming we are on regions where it can be applied safely.

The rest of the paper is as follows. In sections 2 and 3 we describe and exemplify a method for testing the proposed formula on the branch cuts. In section 4 we propose a new method based on Riemann surfaces to make testing the identity on all cells still more efficient. Finally we summarize our contribution and consider future directions.

2. COMPUTING THE ADHERENCE

If we define $\sqrt[0]{p(z)} = p(z)$, then we may define $\mathbb{C}_{rt}(z)$ recursively to be the set of functions of the form $\phi(\sqrt[n]{p_i(z)})$ where ϕ is a rational function with $p_i \in \mathbb{C}_{rt}(z)$ and $n \in \mathbb{N}$. By a *base (inverse) function* we shall mean any of the 14 *inverse* elementary functions, such as $F(z) = \arcsin(z), \log(z)$. These are the only elementary functions with branch cuts, so they will be the main focus here. It will be convenient to work with functions of the form $H_i = F_i(G_i(z))$ where $G_i \in \mathbb{C}_{rt}(z)$ and F_i is an n^{th} root or logarithm as our 'building blocks'; since all of the base functions can be defined in terms of these, it follows that all our admissible formulae (as in 1.1) can be expressed recursively in terms of the H_i. The definitions we use for the base functions are those from [12], although everything which follows applies regardless of the initial choices made, as long as we are consistent.

With F as above, the Riemann surface associated with F shall be denoted by $R_S(F)$, and we recall that this is a path-connected domain for F having either n or an infinite number of sheets respectively, each of which is a domain for a particular branch of the function in question. The branches for $F(z)$ in this case are of course either $\sqrt[n]{g(z)}\exp(\frac{2k\pi i}{n})$ for $k = 0, \ldots, n-1$ or $\log(z) + 2k\pi i$ for $k \in \mathbb{Z}$. For a fixed k, we shall refer to a particular k-th branch of F, denoted by $f_{\_k}$ which has domain which will refer to as comprising the k-th sheet of $R_S(F)$; we denote the principal branch of F by $f_{\_0}$, or simply f. Further, $R_I(F)|_c$ shall be the *Riemann index* of F on a particular cell c of the CAD induced by the branch cuts: that is, either $\exp(\frac{2k\pi i}{n})$ or $2k\pi i$ for some fixed k. The branch cuts serve of course, to act as boundaries where distinct sheets are joined; we denote the set of branch cuts of f by $B(f)$. The important question we shall address is, which sheet does a given branch cut belong to? We now make a key definition, for the case of CADs in the plane.

DEFINITION 1. *(Adherence)*
Suppose that we have a CAD for $B(h)$, and c is a section of a stack representing part of, or all of, a particular branch cut. Let s be an adjacent sector cell to c. Then we say that the branch cut c adheres *to s if c belongs to the same sheet of $R_S(H)$ as does s.*

Recall that, two cells of a CAD are said to be *adjacent* if their union is path-connected[11]. When $H = F(G)$, c can not adhere to both adjacent cells by monodromy.

A brief overview of the algorithm, for the single $\mathbb{C}$ variable case, is presented below. Its purpose is to solve the problem of testing an identity on a branch cut by using an adjacent cell *of full-dimension* instead. In what follows, for any $x \in \mathbb{R}$ we define $\operatorname{sign}(x) = 1$ if $x > 0$ and $\operatorname{sign}(x) = -1$ if $x < 0$. Given an input formula ϕ, we recall that the TruthValue of a cell c in the CAD with respect to ϕ is a boolean value, depending on whether or not $\phi(p)$ is satisfied at any $p \in c$.

ALGORITHM 1.
Input: $H = F(g)$ where F is Log *or* ${}_n$Sqrt., $g \in \mathbb{C}(z)$ [2]
Output: A CAD of $B(h)$; adherent cells determined.

```
1. Compute $S = B(h)$ and $D = CAD(S)$
2. Compute $\operatorname{adj}(D)$ if need be
3. For each $c \in D$ where $TruthValue(c) = True$ do
    sign := $\operatorname{sign}(\Im(g(p)))$, $p \in c_1$, with $s_1, c$ adjacent
    if sign = 1 then $R_I(H)|_c = R_I(H)|_{s_1}$
    else $R_I(H)|_c = R_I(H)|_{s_2}$ where $s_2$ is adjacent to $c$ and not to $s_1$.
```

We now comment further on this algorithm. The role of steps 1 and 2 is obvious. The adherence of any branch cut

[2]The case of $G \in \mathbb{C}_{rt}(z)$ has extra difficulties which we return to later in sections 2.1, 3.

of $f(g)$ will depend only on what we what choose to be the closure of the base function $f(z)$. Examination of the definitions of the complex logarithm and n^{th} root functions shows that this in turn is determined only by what definition we take for the principal branch of arg: here we choose the modern convention which requires that $\arg(z) \in (-\pi, \pi]$. Thus for the branch cut of $f(z)$, we have Counter Clockwise (CC) Closure, or, in the terminology here, the branch cut c of $f(z)$ adheres to the cell T having cell index (1,3), which lies in the same stack in which c does in the CAD constructed with respect to this single branch cut. We now make step 3 more precise. We shall require that s_1 is of full-dimension. Thus in the x—y plane, there are two possibilities for c: either (a) c is a vertical line $x = a$ for $a \in \mathbb{R}$ or (b) it is not. In the former case we choose a sector cell s_1 which is adjacent to c and lies in either of the two adjacent stacks to the one in which c resides, (another reason for treating these cuts specially is given in 2.1) and in the latter case, we choose one of the two sector cells that are adjacent to c in the same stack. Notice that for two-dimensional CADs such as here, we can decide `a priori` whether intra-stack adjacency alone, which does not require an algorithm, will be sufficient; thus allowing us to bypass step two above. To do this we examine the CAD data structure to see if there are any cells which have truth value $True$, are of dimension one, and are constructed over level one cells of zero dimension. The idea behind step 3 is the following. Let $\gamma : [0, 1] \to \mathbb{C}$ be a path with $\gamma(0) = p$ and $\gamma(1) = q$, with $q \in c$, and let $g(q) = s$ where $s \in \mathbb{R}^-$. Then if $g(\gamma(t)) \to s + 0^+ i$ as $t \to 1$ then c adheres to s_1; otherwise, it must adhere to another cell, s_2 say. In case (a) this s_2 will be an adjacent sector cell to c, but in the different stack to which s_1 resides; in case (b), it will be the only other adjacent sector to c in that stack.

Applying an adjacency algorithm[11] to D produces output which can be thought of as comprising lists of cell indices where all indices in a list represent cells which form a single connected component. Computing the adjacencies thus gives us the cells s_1 whose sample points are required for the last step. Notice that there will be any finite number of adjacent cells s_1 to c in general. We only need to choose one cell of course for each branch cut that comprises a single connected component, and this can be achieved by using the adjacency information, thus avoiding the potential redundancy. In the case where $g \in \mathbb{C}(z)$, the particular choice of adjacent cell does not matter.

We point out that the representation used in step 1 will produce a CAD containing other sections which are not branch cuts, and so we do not wish to apply the algorithm to them. For example, if we wish to represent a semi-circular branch cut we would use $\{(x^2 + y^2 - r^2 = 0) \wedge (x > 0)\}$, but this will make for a CAD containing *both* roots of the bivariate polynomial. However these unwanted cells will always have TruthValue $False$, so can be ignored.

2.1 Nested roots

Suppose that $H = F(G)$ where $G \in \mathbb{C}_{rt}(z)$. In [4] we pointed out (for the case of square roots) that the method to calculate $B(h)$ will produce a semi-algebraic set S say that contains *spurious* branch cuts. Recall that in the simplest case, when $g = \sqrt[n]{p(z)}$, that these are sets $\{z \mid (g_{\_k}(z) \in B(f)) \wedge (k \neq 0)\}$— although in general g may contain several n^{th} roots. They arise as an artefact of the method to remove n^{th} roots from the input formula. The problem is that in the CAD solution formula construction step, they comprise cells which are assigned the truth value $True$, since they satisfy the formula S. In some cases one can evaluate $s = g(p)$ for p on the sample point of each cell in S and check whether or not $s \in B(f)$. In general, we run into the same problems as described above in (1.1). One might think that we could detect the spurious cuts by examining the signs of the imaginary parts of $g(p_i)$ for $p_i \in s_i$ with $i = 1, 2$, but the fact that we cannot guarantee that $s \in B(f)$ means that the reasoning based on the continuity of g is not sufficient.

For the important case of *square* roots, we showed in [4] how to remove the spurious branch cuts by adding polynomial constraint equalities to the system. It is a minor extension to handle n^{th} roots which we defer to [6]. However this contributes exponentially to the number of polynomials that are used in the CAD construction. A further difficulty is that the faster methods for eliminating roots such as Gröbner bases, or better, the method of [5] work as black-boxes and do not *automatically* generate the appropriate sets of constraints at each stage. Except in simpler cases, where it may prove to be efficient to remove the spurious cuts, we propose to apply algorithm (1) exactly as we did before; this can be done since for any $p \in s_1$, $\Im(g(p)) < 0$ or $\Im(g(p)) > 0$. (In the case where $\Im(g(p)) = 0$, we know the cut is spurious) Suppose that, as in step 3 that c is in fact spurious, although we cannot yet decide if this is so. It is easy to see that s_1, s_2 will then belong to the same sheet of $R_S(H)$ as does c, and so correctness at the sample point testing stage is guaranteed. The 'adherence information' we obtain in this case is of course spurious but since c adheres to *both* the s_i, we cannot obtain incorrect results. Whilst this approach provides us with a generic algorithm, it is wasteful in that we must compute $g(p)$ for each spurious c nevertheless. One further issue to be exemplified later in section 3, is that given $f(g)$ it may happen that a (non-spurious) cut c derives from both f and g. In that case, we must first calculate the adherence of c with respect to g, before we can compute the adherence with respect to f. This is because, unlike for the case where $g \in \mathbb{C}(z)$, the choice of s_1 in step 3 does matter: we need to choose it so that c adheres to s_1 with respect to g, and then g will be continuous onto the c, as required.

2.2 Justification

In the interest of brevity a detailed proof of correctness shall not given in this paper. However the essential ideas can be sketched as follows. First, one must remember when computing with g in the manner above that g is a *complex* valued function, $g : \mathbb{C} \to \mathbb{C}$, only the branch cuts are real algebraic. However working with both real and complex geometry in this way does not cause any problems provided that one stays away from regions of non-analyticity. (This is why we do not allow complex conjugation: if $g(z) = \overline{z}$ for example, then one can easily check that this does not satisfy the Cauchy-Riemann equations, and is therefore not analytic.) We have that $g(p) \in B(f)$ by necessity. Now g is continuous on a path from at least one of the p_i in the adjacent sectors onto p since in following [4], at step 1, we always include the branch cuts at infinity: that is where the denominator of g vanishes; the only other potential problem would be discontinuities arising in g due to the presence of n^{th} roots, (as in example 3, next section) but as with the singularities they will be by virtue of the CAD, not inside the adjacent sectors we are looking at. The only case where this does not occur is when, given $f(g)$, the cuts for f and g coincide, but this situation is not a problem if we deal with it as described at the end of (2.1). In the case of vertical line cuts, the end points may be singularities. However, since we use sectors in adjacent stacks, as opposed to using the sector above and below such cuts, we avoid losing the notion of a continuous path onto the cut.

We remark that [11] is the most recent adjacency algorithm at the time of writing; we should mention that this may fail in higher than four dimensions, albeit with a very low probability. In the case of failure, we cannot guarantee to be able to represent the section in question correctly and we may under-represent the branch cuts. Fortunately for us, the CADs derived from formulae we are most interested in have at most 4 dimensions.

2.3 Application

In general the input formula $H = 0$ contains several of our $H_i = F_i(G_i)$ building blocks. To deal with this, all that one needs to add in an implementation is a piece of additional information, namely, which H_i each branch cut cell derives from, to the usual CAD cell data structure. (Such as that found in the package we have used here[24].) that is, which H_i each branch cut cell derives from. Then we should simply apply the algorithm as shown, only using this data to ensure that we apply the appropriate g_i at stage 3. Of course, some branch cut sectors may derive from several H_i. Now the adherence information alone is insufficient to determine the actual $R_I(H_i)$ on a cell. However the point is that we now have a much simpler method (modulo the remark at the end of this subsection) to determine the truth of *the formula* $h \overset{?}{=} 0$ on the branch cuts by testing the appropriate adjacent full dimensional cells instead. Apart from the possibility of choosing unlucky sample points (this will be avoided using the method of section 4), it is now an easy task to calculate the correction factor for $h = 0$ on these cells by using the method of [8] which uses floating point evaluation of H at the sample point together with bounds on the accuracy derived in the manner shown there. Of course this does not tell us what $R_I(H_i)$ is for individual H_i. A problem occurs however, on branch cut cells which derive from several H_i, for some of these functions may adhere to different cells. An example of this is discussed in section 3.

It is often convenient for a user to specify a function by arccosh say, as opposed to giving its definition in terms of logs and square roots. We have therefore determined the adherence for each of the base functions using the method above so that one can now allow our input function F to be any of these 14 functions. For future reference, this information is presented in the appendix. If a particular cut of F under consideration lies on the imaginary axes of the $\mathbb{C}$ plane, then one must make the necessary minor changes in step 3: we calculate $sign := \operatorname{sign}(\Re(g(p)))$ instead for cuts that lie on the imaginary axes of the $\mathbb{C}$ plane, and then if the cut is from the side $\Re(z) < 0$, for example, (as given in the table) the condition should be taken to be $sign = -1$.

3. EXAMPLES

The three simple examples which follow are chosen so that they may be readily verified by the reader. In the first two examples, x and y shall be real. The case of $\mathbb{R}^2$ will be seen to exhibit different behaviour from that of $\mathbb{C}$, for we now

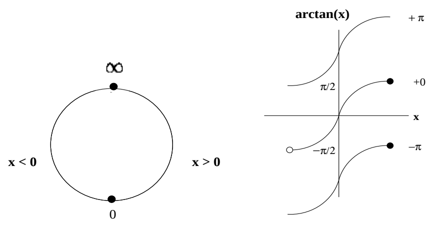

Figure 1: $\mathbb{R}^*$ & corresponding branches for $\operatorname{Arctan}(x)$**.**

have branch cuts at infinity (example 1) and singularities (example 2) which are not just zero dimensional regions.

1. $H = \operatorname{Arctan}\left(\frac{x+y}{1-xy}\right) - \operatorname{Arctan}(x) - \operatorname{Arctan}(y) = 0;$

we now investigate the corresponding identity $h \overset{?}{=} 0$, where $h = h_1 - h_2 - h_3$. For real inputs, $\arctan(x)$ only has branch cuts at $\pm\infty$. Our representation is only for finite branch cuts, so we see that the branch cuts for H derive from h_1 only: that is, the set $B(h) = \{xy - 1 = 0\}$. This comprises the two branch cuts shown in figure (2) which now require investigation. By numeric or symbolic evaluation, the identity is readily verified to be true on region 2. On region 1 we have that $H = -\pi$ and that $H = \pi$ on region 3. We now consider the formula on the cuts themselves. Notice that the CAD data structure will tell us that we do not need to compute the adjacency of the CAD with respect to $B(h)$ here. Clearly, a purely numeric approach here would fail, at any point on the cuts, due to the blow-up. We shall show that $\arctan(x)$ must be viewed as a function of the form, $(-\infty, \infty] \to (-\frac{\pi}{2}, \frac{\pi}{2}]$: that is the point $(\infty, \frac{\pi}{2})$ belongs to the principal branch of arctan whilst the point $(-\infty, -\frac{\pi}{2})$ belongs to the branch, $\arctan(x) - \pi$. This requires us to work on the extended real line $\mathbb{R}^* = \mathbb{R} \cup \{\infty\}$ which we recall is constructed by identifying the end points $\pm\infty$ as in figure (1).[3] Thus on $\mathbb{R}^*$ one works with *positive infinity* only,and if $x \to \infty^+$ then we see that one passes *onto* the point ∞ continuously (see the left-hand diagram of figure 1) whilst if we let $x \to \infty^-$ then we do not. In order to preserve the continuous 1-1 correspondence between the finite domain of $\arctan(x)$, that is $\mathbb{R}$, and the extended domain that is $\mathbb{R}^*$, we see that $-\infty$ does not belong to the principal branch domain of $\arctan(x)$. As always, when passing through a branch cut one passes onto the adjoining branch domain; we can see that in this case that this will be the branch $\arctan(x) - \pi$, as in the right-hand diagram of figure (1).

Let $c_1 = \{(xy - 1 = 0) \wedge (x > 0)\}$ and $c_2 = \{(xy - 1 = 0) \wedge (x < 0)\}$, and put $g = \frac{x+y}{1-xy}$. Now we can use adherence to determine which branch each c_i is on by determining whether g tends to $+\infty$ or $-\infty$: the c_i will adhere to the cell where g is positive. This requires that our adjacent cells are sign-invariant for g so we must first add the line $y + x = 0$ to the CAD. Consider c_1, where our adjacent regions are 1 and the part of 2 above the line, and then suppose that our sample points for these are given by $p_1 = (1, 2)$ and $p_2 = (1, \frac{1}{2})$ respectively. We obtain $g(p_1) = -3$ and $g(p_2) = 3$, which shows that c_1 adheres to region 2, and we conclude the identity is true on this branch cut. Similarly we treat $c_2 = \{(xy-1 = 0) \wedge (x < 0)\}$ using regions 2 (below the line) and 3 and $p_3 = (-1, -2)$ and $p_2 = (-1, -\frac{1}{2})$. This

[3]Corresponding to a circle on the Riemann Sphere.

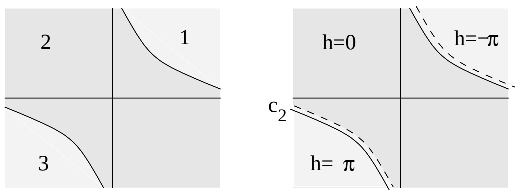

Figure 2: decomposition & adherence for Ex.2.

time $g(p_3)$ is positive and $g(p_2)$ is negative, so c_2 adheres to region 3. Thus we conclude that the identity is false on this branch cut, and requires a correction factor of $-\pi$.

Observe that the example could have been written in terms of complex logarithms. Then we could proceed by using the fact that $\log(g) = \log|g| + i\arg(g)$ where $\arg(g) = \arctan(\frac{\Im(g)}{\Re(g)})$. Since the argument component determines which branch we are on, and the input to the arctan is wholly real, the problem then reduces to the previous case. We omit the details for brevity. This approach should allow us to consider more general examples, although this is work in progress.

2. $H = \mathrm{Log}(xy) - \mathrm{Log}(x) - \mathrm{Log}(y) = 0$.
The branch cuts for the first logarithm are the top-left and bottom right-hand quadrants of $\mathbb{R}^2$, together with the x and y axes. The latter set comprises the singularities; notice that they are one dimensional, and that they disconnect the plane— this has implications for the algorithm in section 4. As is well known, at the point of ramification that is $z = 0 \in \mathbb{C}$, all the sheets of $R = R_S(Log(z))$ (and also for the n^{th} root) are joined, but this point does not actually belong to R, corresponding to the fact that $\exp(z) \neq 0$ for all z. So unlike the previous example, it does not make sense to ask which one of adjacent sheets of $\mathrm{Log}(z)$ does this point adhere to. The branch cuts for the other two logarithms are immediately clear, and it therefore turns out that we only can test the identity on the branch cuts. Since these are of full dimension, there is no problem: evaluating $\log(xy) - \log(x) - \log(y)$ at (-2,3),(2,3) and (2,-3), for example, shows that the H is always true on the branch cuts and undefined everywhere else.

3. $\mathrm{Arcsin}(z) \subset \mathrm{Arctan}\left(\frac{z}{\mathrm{Sqrt}(1-z^2)}\right)$.
Notice we only have set inclusion, not equality. We now consider the often quoted 'identity': $\arcsin(z) \overset{?}{=} \arctan\left(\frac{z}{\sqrt{1-z^2}}\right)$. Let $z = x + iy$. Writing $\arcsin(z)$ in terms of its definition, $-i\log(\sqrt{1-z^2} + iz)$, let us label this as $h_1 \overset{?}{=} h_2$ with $h_i = f_i(g_i)$ for $i = 1, 2$. Following the branch cut finding algorithm we see that each $g_i(z)$ has branch cuts given by $c_1 \vee c_2$ where $c_1 = \{(x < -1) \wedge (y = 0)\}$ and $c_2 = \{(x > 1) \wedge (y = 0)\}$. For the branch cuts of h_2 we obtain the three-dimensional semi-algebraic set,
$\phi = [(y^2 - x^2 + 1 \leq 0) \wedge (y = 0) \wedge (y^2t^2 - x^2t^2 + t^2 - y^2 + x^2 = 0)] \wedge [(t + 1 \geq 0) \vee (t + 1 \leq 0)]$.
As described in [5] we must now apply Quantifier Elimination (QE) to the formula $(\exists t)\phi$ in order to eliminate t and obtain a formula without this extraneous variable. This yields the set comprising ${c_1}' \vee {c_2}'$, which is in fact identical to $c_1 \vee c_2$. Now h_1 does not have any branch cuts since (by QE) g_1 never maps any points onto the branch cuts of $f_1(z)$. We now calculate the adherence of h_2 for the branch cut ${c_i}'$. Following (2.1), we must first calculate the adherence of the c_i (with respect to g_2). This is easily verified by algorithm 1 to yield the adherence of c_1 to the region s_1 containing the point (-2,1) for example, whilst the cut c_2 adheres to the region s_2 containing the point (2,-1). We must therefore choose these points with which to compute the adherence of the ${c_i}'$. Comparing the results to those of h_1 (from the table) we see that the adherence of the cuts for h_1 is different to the adherence of the cuts for h_2.

The problem arises if we try to apply this adherence knowledge *directly*: one clearly cannot test the identity on either of the cuts by using any single sector point! When are testing $h_1 \overset{?}{=} h_2$ and the adherence differs in this way on a cut, we should test the identity on the adjacent regions first. Then if we find a adjacent sector where the identity holds, then this is sufficient to conclude that the identity cannot hold on the cut itself. In this example, the identity easily proven to be true everywhere off the branch cuts, and so we conclude that it is false on the c_i. To see this, we argue as follows. Suppose that we take a continuous path onto a point $p \in c_1$ from s_1. Now $h = h_1 - h_2$ is analytic and $h = 0$, until we reach the end-point of our path, whereupon h_2 *only* becomes discontinuous, and so $h(p) \neq 0$. Hence by the Monodromy theorem, $h \neq 0$ on c_1. The argument for c_2 runs entirely analogous.

4. WINDING SEQUENCES

In [22] it is reiterated that consideration of Riemann surfaces might yield an effective algorithm and a request is made to the community for further investigation into this area. Some of the difficulties with making these objects computational was pointed out in [13]. We do not propose a new representation for computing with Riemann surfaces; rather, we propose to compute with the well-known, theoretical 'cut-plane' representation of Riemann surfaces. To make this computational, one needs to bear the structure of the original surface in mind as it is projected down onto $\mathbb{C}$.

The prevailing ethos behind this approach is to capture the idea of a continuous choice of argument whilst retaining the geometry and sense of where one is on the Riemann surface; as such this method can be viewed as an effective synthesis of the work reported in [13] and the Decomposition Method. A major problem with the cut-plane approach is the multiplicity of representation: informally, each region represents more than one (and in the case of logarithm, infinitely many) sheets of the Riemann surface. A step towards fixing this problem lies in the following definition. Let $H = F(G(z))$ be as in section 2, and Φ a formula, although we shall not allow nested roots for the moment.

Definition 2. *(Reachable Sheet)*
The k-th reachable sheet *of $R_S(H(z))$ is the sheet with index k such that there exists $z \in \mathbb{C}$ such that* $\mathrm{Arg}(g(z)) \in M_g$, *where $M_g = ((2k-1)\pi, (2k+1)\pi]$; we refer to the set of all such sheets as* reachable sheets, *denoted by $\hat{R}_S(H)$.*

We define the *reachable cells* of a $R_i \in \hat{R}_S(H)$ analogously. We may compute M_g by factorising numerator and denominator into the form $\prod_j \alpha_j \prod_i (z - \alpha_i)$ for $\alpha_{i,j} \in \mathbb{C}$ and using the multivalued rules for Arg. For example, given $h = \log(z - z^2)$ we obtain $\mathrm{Arg}(z - z^2) = \mathrm{Arg}(z) + \mathrm{Arg}(-1) +$

$\mathrm{Arg}(z-1)$ and then $(-\pi,\pi]+\pi+(-\pi,\pi]=(-\pi,3\pi]$. Whence there are only 2 reachable sheets, having indices 0 and 1, although $R_S(H)$ comprises infinitely many sheets. The number of reachable sheets of $R_S(H)$ is always finite, and $\hat{Z}\subset\mathbb{Z}$ shall be the set of reachable sheet indices.

With $H=F(g)$ being one of the building blocks in a formula, we shall demonstrate that, if we construct a CAD with respect to a set S to be described below, then we can obtain a surjective correspondence ψ from the set of regions of D onto $\hat{R}_S(H)$. That is, each cell in the CAD corresponds to part of, or all of a unique element of $\hat{R}_S(H)$. ψ is not a bijection because one of the draw-backs of CAD is that of over-representation; it generally produces far more cells than the actual number of connected components of the decomposition induced by the algebraic curves. With g factored into linear factors p_i as above we compute $S(h)=\{B(f(p_i)),\, L(f(p_i)), B(h), L(h)\}$ for each i where we define $L(\tilde{h})$, the set of *argument branches* of $\tilde{h}$, to be the set of points $\{z \mid \tilde{g}(z)\in[0,\infty)\}$. We shall later require the *principal* argument branch, denoted by $L_0(\tilde{h})$: this is the set such that $\mathrm{Arg}(p)=2k\pi$ for all $p\in L_0(\tilde{h})$, where $|k|$ is the least element of $\hat{Z}$. We compute the semi-algebraic set representing the argument branches by using exactly the same algorithm as we did for the branch cuts.

We now make the key definition of this section. Given H as above suppose that we have computed $S(h)$ and that we have constructed a CAD D with respect to this set.

Definition 3. *(Winding Sequence)*
A Winding Sequence *(WS) for D shall be a finite set of ordered lists called* paths l_i*, each of them finite, and where each element of the list is a cell index c_i for $i\in 0,\ldots,N_{l_i}$ and the following conditions are satisfied:*

1. *c_i, c_{i+1} represent adjacent regions on $\hat{R}_S(H)$;*
2. *each cell in D has an index belonging to at least one of the l_i;*
3. *c_0 represents L_0, or part of L_0 in at least one of the l_i.*

Before we give an overview of the proposed algorithm, we need to make one more definition. The *index aggregate* of $\Phi=\phi(H_i)$ on a cell c, denoted by $I_A(\Phi)|_c$ shall be the result of making a set of substitutions [4] into Φ of the form, $F_i\to f_{i\_k_1}$, and for each subexpression G_j of G_i where $G_j={}_nSqrt(.)$, $G_j\to g_{j\_k_2}$ for some fixed $k_1,k_2\in\hat{Z}$.

Algorithm 2.
Input: $\Phi=\phi(H_i)$
Output: A CAD D, $I_A(\Phi)|_{c_k}$ for all $c_k\in D$
For each H_i do steps 1-3

```
1. compute $\hat{Z}$
2. Compute $S(h)$, $D = CAD(S)$ and adj$(D)$.
3. choose $l_i$ until every cell of $D$ where $H_i$ is defined and
   has been included in at least one of the $l_i$
4. For each cell $c$ compute and simplify $I_A(\Phi)|_c$.
```

The above computes paths through the cells of the CAD which correspond to paths on the Riemann surface of each H_i in turn. It is important however to notice that ensuring the adjacency of cells in a path is a necessary, but not sufficient condition for this to be the case. We point out that computing a separate path for each H_i effectively side-steps the problem raised in [13] of what to do when one has a *sum* of H_i such as H_1+H_2 where the H_i are two separate Riemann surfaces.

First we must locate which sections derive from the $L_0(H_i)$ and choose one to obtain the initial cell of our path, c_0. Let us compute $M=M_g(g_i)$. There may be several argument branches for any number of the reachable sheets, but the L_0 branch(es) may be successfully singled out by computing $\mathrm{Arg}(g(p_i))$ where p_i for $i=1,2$ are the sample points of two adjacent sectors to an L_i. We do this by factorising g as we did to find M_g above and the argument of each factor is computed by floating point evaluation. We now try to determine $R=R_I(c_0)$. As we see in (4.1.1), we may have that $R\neq 0$, since $0\notin M_g$. In this case, c_0 (nor any other cell in L_0) cannot be our first cell in the WS and we must leave it until it is reached later in the path, or in another path. In this case, only one of the adjacent sectors c_l and c_k say, to c_0 can be chosen as the next cell in the path. Again we must examine M_g to decide which.

Suppose that we have now reached a sector cell c_{j-1} in our current path, and determined $R_I(H_i)|_{c_{j-1}}$. It remains to show how to choose a section c_j so that we still have a WS. Let us call the sections that derive from the set $S(h_i)$ under consideration *critical sections*; the current $R_I(H_i)$ can change only if c_j is one of these. There are 2 cases; first suppose that $s=c_j$ derives from $B(h_i)$. Then the adjacency of the CAD of S, computed at step three, allows us to apply algorithm (1): we outline the few possibilities. If $H_i=\mathrm{Log}(g(z))$ then if we approach s from the non-adherent side then we subtract $-2\pi i$ from $R_I(H_i)|_{c_{j-1}}$ to obtain $R_I(H_i)|_s$, and then $R_I(H_i)|_{c_{j+1}}$ has the latter Riemann index for our next valid cell in the path. If we approach s from the adherent side then we add $2\pi i$ to $R_I(H_i)|_{c_{j+1}}$ only. If $H=\sqrt[n]{g(z)}$, we either reduce or increase $k\in\mathbb{Z}_k$ by 1 in the Riemann index $\exp(\frac{2k\pi i}{n})$ each time, noting that if $R_I(H_i)|_{c_{j-1}}=1$ and if c_{j-1} does not adhere to s then $R_I(H_i)|_s=\exp(\frac{2(n-1)i\pi}{n})$. This is efficient as no evaluation of Φ need be done on these sectors. However, if $R_I(H_i)|_{c_j}$ corresponds to a non-reachable sheet (which can be checked since we already know $\hat{Z}$) then we must in fact terminate the path after or before including c_j in the path, depending on the adherence of s. The second case occurs if c_j derives from any of the other functions in $S(h_i)$. Then, we may also have the same problem as just described, or more difficult to detect, it may happen that $R_I(H_i)$ changes on c_j even though it is a branch cut or argument line of *one (or more) of the* $f(p_i)$. Thus we may obtain the wrong index if we simply include c_j as our next cell in the path. In this case, there are two approaches. The first is to try to find new paths avoiding c_j starting from the last valid cell in the former path. The second requires us to compute $\mathrm{Arg}(p)$ on the sector c_{j+1} by the M_g method to give us the correct $R_I(H_i)|_{c_{j+1}}$ and then use adherence to determine the index for the section c_j.

The last step, where we simplify $I_A(\Phi)|_c$, is required in order to decide whether the correction factor for the single-valued formula $\phi(h_i)=0$ is zero or not on each cell. For non-nested g_i it is just a case of collecting the Riemann indices from the logs together, treating π as a symbolic indeterminate, and similarly for the roots. The nested case is a little more involved and will not be described here but in both cases the general constant problem does not arise.

[4] A similar notion used in a different context occurred in [19].

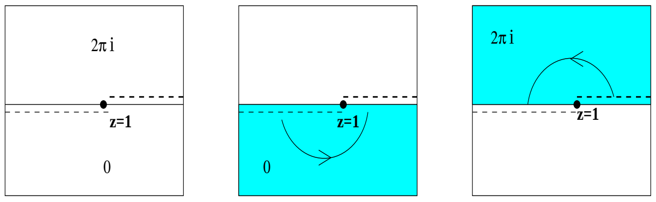

Figure 3: The reachable sheets of $R_S(\text{Log}(1-z))$.

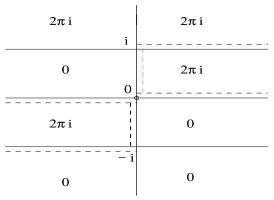

Figure 4: Riemann indices for $\text{Log}(-1-z^2)$.

4.1 Examples

1. $H = \text{Log}(1-z) - \text{Log}(-1) - \text{Log}(z-1)$.
The sets of indices for the reachable sheets corresponding to the three logarithms above are $\{0,1\}$, $\{0\}$ and $\{0\}$ respectively. Thus we only need to compute a WS for the $\text{Log}(1-z)$, as this is the only one which will affect $I_A(H)$. Clearly $M_g = (0, 2\pi]$. The branch cut and argument branch for this first logarithm are given by $[1,\infty)$ and $(-\infty, 1)$ respectively, and the adherence of these can be easily calculated to obtain the set-up shown in the left-most diagram of figure (3). The other two diagrams show $R = \hat{R}_S(\text{Log}(1-z))$, namely domains of the principal and $2\pi i$ branches, and the shaded regions represent the reachable cells of R together with their Riemann indices. (The simple CAD comprises the 9 obvious cells but is not shown here for clarity) We now show to derive those values so that they may be used to anotate the cells in a way that matches the left-hand figure as shown.

There is only one argument branch here so this must be L_0, and so we let this cell be c_0. However, the Riemann index of c_0 cannot yet be determined, as $0 \notin M_g$. Its cell index is (1,2) from which adjacency gives us a choice of (1,3) or (1,1) or (2,2) for c_2. M_g tells us that we must identify the cell which has points such that $\text{Arg}(g)$ is positive, and so using $p = -1+I$ for example, we see that $sign(\Im(g(p))) < 0$ showing that (1,1) is a reachable cell *on the principal sheet*, with Riemann index 0, whilst (1,3) is not; (2,2) is a singularity, and cannot belong to the WS either. Now we take $c_3 = (2,1)$ and $c_4 = (3,1)$, for which the index remains 0. Next, $c_5 = (3,2)$ is identified by the data-structure as a branch cut, and algorithm (1) shows that the index should be 0 on it and $2\pi i$ on the next cell $c_6 = (3,3)$. Finally the remaining steps of the WS are (2,3),(1,3), and (1,2) where the index remains at $2\pi i$, and we discover that c_0 in fact belongs to the sheet with index 1.

2. $H = \text{Log}(-1-z^2) - \text{Log}(i-z) - \text{Log}(i+z)$.
The sets of indices for the reachable sheets of each logarithm are $\{0,1\}$, $\{0,1\}$ and $\{0\}$ respectively, so we need to compute a WS for the first two logarithms shown. For brevity, we just focus on the first logarithm with the others being similar to (4.1.1). We compute the M_{g_i}, argument branches a_i and branch cuts b_i for each logarithm. For the first one, L_1, we obtain $(-2\pi, 3\pi]$ and the line segments $a_1 = (i, i\infty)$ and $a_2 = (-i, -i\infty)$, (an easy computation shows that this is L_0) as well as $b_1 = (-i, i)$, and b_2 being the $x-axis$. Figure (4) shows what $R_I(L_1)$ is alone on each of the cells. Now a possible set of paths are as follows. $l_1 = \{(2,1),(1,1)\}$, $l_2 = \{(2,1),(3,i),(2,7),(1,7),(1,6)\}$ where $i = 1, \ldots 7$ and $l_3 = \{(3,3),(2,3),(1,3),(1,j)\}$ where $j = 2, \ldots 5$. Now l_1, l_2 were terminated on finding a cell where the $R_I(L_1)$ cannot be found without deducing, via adjacent cells, the value of $Arg(p)$ with $p \in L_1$ at its sample point. The l_i are efficient paths which complete most of the WS without performing these computations. We point out that from M_g we may determine $R_I(L_1)$ on (1,1) and (3,1) for free as well as (3,2), even though the latter cell is a critical section deriving from $F(p_i) = \text{Log}(i+z)$, and would generally require an argument calculation. Note that after finding (1,3), (1,2) only requires the adherence of the b_i which is this cell. Finally, the points $(0, \pm i)$ but not (0,0), can be placed into the above WS in the obvious way.

4.2 Remarks

A major motivation for presenting algorithm (1) as a special case of algorithm (2) is due to the extra computational cost involved in steps 1 and 2; particularly the latter as we effectively double the number of polynomials in our semi-algebraic set. Fortunately the cost of CAD is doubly exponential only in the number of variables[17], and is polynomial time in the number of polynomials in the input formula.

If we are to apply (4.1) to examples where the $G_i(z)$ are to admit n^{th} roots, then we work recursively, and we use the fact that $M_g(\sqrt[n]{p(z)}) = \frac{1}{n} M_g(p(z))$. However we require that the inputs to the roots (recursively) are single ‘root monomials’, as defined in [4], only. In this case, paths must terminate on or before encountering branch cuts deriving from an n^{th} root, depending on their adherence. The general case remains to be done.

An another advantage of WS is that they solve yet another problem, not mentioned in (1.2) that occurs in the sample point testing phase. Numerical evaluation will produce in most cases an arbitrary looking floating point number. What would be more useful would be to determine a symbolic correction factor for the proposed formula $H = 0$ where required. As in [8] one can try to determine this by calculating error bounds and choosing the branch with the closest value at p although this is not always possible. It also requires that we prohibit inverse elementary functions appearing in denominators[5] for we need to ensure separation of the set of values obtained by evaluating each branch at the point p. Working symbolically will be better suited to achieving this goal, although this will be more costly in general, and it will not reveal what branch we are on for each of the *individual* H_i in Φ. WS do provide this information, and it may be of great interest to the user.

The cost of accessing the data-structure is the only price we pay when determing the Riemann index on many of the cells. Thus, the loss of efficiency caused by the over-representation of D mentioned earlier should be somewhat offset by the gains made here, without the need for clustering. This should be useful for cases where our CAD comprises several thousand cells[6].

[5]See [8] for examples.

[6]This is not unusual, see [5] for examples.

5. CONCLUSION & FUTURE WORK

We have seen that the adherence method in section 2 deals with some of the more difficult issues associated with the Decomposition Method in an efficient way. The only drawback in terms of complexity is that we may have to use an adjacency algorithm. It would still be nice to implement the method so that we may attempt larger examples and see what sort of run times we obtain. The Winding Sequence approach avoids the problems of unlucky sample points, the constant problem and models well the underlying cause of the problem of why multivalued formula can fail to hold; it also has good long term potential for larger function fields, provided we can still generate a decomposition with respect to the cuts. Indeed, methods do exist for the larger class of *Pfaffian functions*[21]. Formal justification for *several* $\mathbb{C}$ variables may be harder to achieve however, for both methods. Until such a time it would be interesting to choose examples which can be verified by other means and see if any counter-examples can be found. Although the complexity (at the last step of the Decomposition Method) of producing CADs with large numbers of cells has been reduced by our method, it would clearly advantageous to produce smaller CADs if possible. One approach to this is to obtain a more accurate description of the branch cuts in the first place; we shall describe this later in [6].

6. REFERENCES

[1] Abramowitz, M. & Stegun, I., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. J. Wiley & sons inc., May 1984.

[2] Arnon, D. S., A cluster-based CAD Algorithm. In *J. Symbolic Computation*, **5**(1/2) 189–212, 1988.

[3] Beaumont, J., Bradford, R. & Davenport, J. Better Simplification of Elementary Functions Through Power Series. In *ISSAC 2003* (2003), J.R. Sendra, Ed., ACM, New York, 30–36.

[4] Beaumont, J., Bradford, R., Davenport, J. & Phisanbut, N. A Poly-Algorithmic Approach to Simplifying Elementary Functions. In *ISSAC 2004* (2004). J. Gutierrez, Ed., ACM, New York, 27–34.

[5] Beaumont, J. C., Bradford, R. & Phisanbut, N. Practical Simplification of Elementary Functions using CAD. *In Algorithmic Algebra and Logic* 2005.

[6] —. Effective Simplification of Elementary Functions *in preparation.*

[7] Bradford, R., Corless, R., Davenport, J., Jeffrey, D. & Watt, S. Reasoning about the Elementary Functions of Complex Analysis. *Ann. Math. & Artificial Intelligence* **36**, (2002), 303–318.

[8] Bradford, R. & Davenport, J. Towards Better Simplification of Elementary Functions. In *Proc. ISSAC* (2002), T. Mora, Ed., ACM, New York, 15–22.

[9] Carette, J. Understanding Expression Simplification. In *ISSAC 2004* (2004), Jaime Gutierrez, Ed., ACM, New York, 72–79.

[10] Collins, G. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. *Proc. 2nd. GI Conf. Automata Theory & Formal Languages* (1975), vol. **33** of SLNCS.

[11] Collins, G.& McCallum, S. Local box adjacency algorithms for Cylindrical Algebraic Decomposition. In *J. Symbolic Computation*, **33**(3):321–342, 2002.

[12] Corless, R., Davenport, J., Jeffrey, D. & Watt, S. According to Abramowitz and Stegun. *SIGSAM Bulletin 34*, 2 (2000), 58–65.

[13] Corless, R. M. & Jeffrey, D. J. The Unwinding Number. *SIGSAM Bull.* **30** (1996) 2, issue 116, 28–35.

[14] Corless, R., Gonnet, G., Jeffrey, D., Hare, D. & Knuth, D. On the Lambert W Function. In *Adv. in Computational Mathematics* **5**, (1996), 329–359.

[15] Davenport, J. H., The Geometry of C^n is Important for the Algebra of Elementary Functions., in *Algebra, Geometry, and Software Systems* 207-224,(2003).

[16] Davenport, J. H., MKM from Book to Computer: A Case Study In *MKM 2003* 17–29, A. Asperti, B. Buchberger, J. H. Davenport, Ed., Springer.

[17] Davenport, J. and Heintz, J. Real Quantifier Elimination is Doubly Exponential. *J. Symbolic Computation 5*, (1988), 29–35.

[18] Dolzmann. A, Seidl. A & Sturm. T. *Efficient Projection Orders for CAD.* In *Proc. ISSAC* (2004), Jaime Gutierrez, Ed., ACM, New York, 111–118.

[19] Fateman, R. J. & Dingle, A. Branch Cuts in Computer Algebra. In *ISSAC 1994, ACM Press, New York*, 1994, 250–257.

[20] Grigoriev, D. Yu. & Vorobjov, N. N.Jr. Counting connected components of a semi-algebraic set in subexponential time. *Computational Complexity* **2** (1992) 133–184.

[21] Gabrielov, A. & Vorobjov, N. Complexity of cylindrical decompositions of sub-Pfaffian sets. *J. Pure Appl. Algebra 164* (2001), 179–197.

[22] Jeffrey, D. J. & Norman, A. C. Not seeing the roots for the branches: multivalued functions in computer algebra. *SIGSAM Bulletin* **vol 38** (2004) no. 3, issue 149, 57–66.

[23] Moses, J. Algebraic Simplification, a Guide for the Perplexed. In *Advances in Computational Mathematics 14*, **no.8** (1971).

[24] H. Hong *et al.* Quantifier Elimination by Partial Cylindrical Algebraic Decomposition. `http://www.cs.usna.edu/~qepcad/B/QEPCAD.html`

[25] Richardson, D. Some Unsolvable Problems Involving Elementary Functions of a Real Variable. *Journal of Symbolic Logic* **33**(1968), 514–520.

[26] Richardson, D. How to Recognize Zero. In *Journal of Symbolic Computation* (1994), 24(6), 627–646.

[27] van der Hoeven, J. A new Zero-test for Formal Power Series. In *ISSAC 2002* (2002), T. Mora, Ed., ACM, New York, 117–122.

APPENDIX

The adherence of the base inverse elementary functions, based on the definitions in [12], is given below.

Function	Branch cut	Closure	Branch cut	Closure
arcsin(z)	$(-\infty,-1)$	$\Im > 0$	$(1,\infty)$	$\Im < 0$
arccsc(z)	$(-1,1)$	$\Im > 0$	—	—
arcsinh(z)	$(-i\infty,-i)$	$\Re < 0$	$(i,i\infty)$	$\Re > 0$
arccsch(z)	$(-i,0)$	$\Re > 0$	$[0,i)$	$\Re < 0$
arccos(z)	$(-\infty,-1)$	$\Im > 0$	$(1,\infty)$	$\Im < 0$
arccosh(z)	$(-\infty,1)$	$\Im > 0$	—	—
arcsec(z)	$(-1,1)$	$\Im > 0$	—	—
arcsech(z)	$(-\infty,0]$	$\Im > 0$	$[1,\infty)$	$\Im > 0$
arctan(z)	$(-i\infty,-i]$	$\Re < 0$	$[i,i\infty)$	$\Re > 0$
arccot(z)	$[-i,i]$,	$\Im > 0$	—	—
arctanh(z)	$(-\infty,-1]$	$\Im > 0$	$[1,\infty)$	$\Im < 0$
arccoth(z)	$[-1,1]$	$\Im < 0$	—	—