A Poly-Algorithmic Approach to Simplifying Elementary Functions

James C. Beaumont, Russell J. Bradford, James H. Davenport & Nalina Phisanbut *
Department of Computer Science
University of Bath
Bath BA2 7AY England
{jcb, rjb, jhd, cspnp}@bath.ac.uk

ABSTRACT

Simplification has been long recognised to be a fundamental problem within computer algebra[20]. However, even for the class of elementary functions, it has not been resolved in a satisfactory way.

Algorithms were presented in [5, 3] to solve this problem, and it was seen that both methods had their own strengths and weaknesses. Also, not all functions could be handled by either of the methods alone. The current paper continues this line of development by combining the two methods, and reporting on progress made with the various sub-algorithms involved.

1. INTRODUCTION

The class of elementary functions is recognized to be ubiquitous in applications, and as a result it is vital to have good algorithms for their manipulation. One of the major obstacles encountered when trying to simplify such functions, either by hand or by using a CAS system, is the fact that all the inverse functions such as the square root or arctan are essentially multi-valued. Branch cuts are a well-known device introduced to define regions on which a multi-valued function F, say, is single-valued; for each region one is said to be working with a particular branch of F. Here we note that, as in previous papers, we shall use the notation that terms such log and $\sqrt{\ }$ denote single-valued functions from $\mathbf{C}$ to $\mathbf{C}$. Capitalised variants, such as Log and Sqrt denote multi-valued functions, regarded as mapping $\mathbf{C}$ into sets of values, so that $\mathrm{Sqrt}(z) = \{w : w^2 = z\} = \{\pm\sqrt{z}\}$ for example. The choice of cuts is not unique, and as far as simplification is concerned, what is the best choice to make has a been a matter for discussion[10]. Whatever (consistent) choice we make, the problem is that when we attempt to make a single-valued simplification of a given expression, one needs to know which branches of the functions involved to take; this depends on the inputs to those functions. A well known example is that of simplifying $\sqrt{z^2}$ to z, where we have taken the positive branch of the root. This is in fact incorrect unless we are in the region of the complex plane where $csgn(z)$ equals 0 or 1; elsewhere we must use the negative branch, obtaining $-z$ instead.[1] Replacing the term z^2 by an arbitary function should convince the reader that this problem can be highly non-trivial.

1.1 Previous Work

Most CAS systems have some simplification routines, but they are usually limited to various special cases. To the best of the authors knowledge, one of the most succesful attempts to solve this problem was in [14]. Restricting to the case of one complex variable, it took what we shall call the decomposition approach, that is it

- calculated the set of branch cuts of the proposed identity;
- found a sample point in each of the regions in $\mathbf{C}$ defined by the cuts;
- evaluated the identity numerically using that point.[2]

A protype implementation in Mathematica was given, which was capable of handling many simple examples. Essentially, branch cuts are represented by a triple (r_0, r_1, f) with $r_0, r_1 \in \mathbf{R} \cup \{\pm\infty\}$ and $f : [r_0, r_1] \mapsto \mathbf{C}$. Calculating the cuts of $f(g(z))$ involves computing $\phi(z) = g^{-1}(z)$. The problem with this is that in general one cannot compute a closed form for $\phi(z)$; for example the function $\sqrt{p(z)}$ where $p(z)$ is a polynomial of degree 5 or more cannot generally be expressed in terms of radicals. This problem reoccurs during the sample point finding process, which requires one to compute with $\phi(z)$ and its derivatives. As the authors themselves point out, the proceedure will not represent regions which are contained entirely inside other regions. In [5], a similar approach was suggested, but introduced the use of multi-valued functions in actually producing a proposed simplification, before following the steps above. The result was an explicit reduction to the constant problem[24]. It

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

[1] Here we use the definition which sets $\mathrm{csgn}(z) = 0$ at the origin.

[2] One sample point is sufficient to represent an entire region by the Monodromy theorem[17].

proposed using cylindrical algebraic decomposition[8](CAD) for the second step, and discussed its feasibility. There was also consideration of multivariate examples, and the actual difficulties involved in doing the sample point testing semi-numerically, as well as some possible solutions. In [3], the main aim was to investigate an alternative approach to doing the sample point testing, by applying the method of [25] to the power series expansions of the functions about the chosen sample point. The advantages and disadvantages of the two approaches were summarised in figure 1 of [3]. Other key approaches not based on performing a decomposition of the plane include the use of *unwinding numbers* [11], and *signed zeros*[18]. We refer to [4] for a discussion of their potential, which also considers yet another possibility, that of using Riemann surfaces.

Since not all of the functions of interest could be handled by either of the methods alone, we propose to combine the two methods here. More importantly though, there were several sub-algorithms involved in the methods of [5, 3] and some of these had not been automated; here we fill these gaps and add other optimizations, thus bringing the decomposition approach closer to a working reality. We feel this is important as the decomposition approach seems to be currently the most promising out of those mentioned above.

Specifically, the contributions from the rest of the paper are as follows. In section two, we outline a method to calculate the set of branch cuts for our chosen class of functions. The representation we use, essentially sets of polynomial inequalities, allows us to handle a wider class of functions than [14], as well as allowing us to make use of CAD, and its powerful variants, such as *partial* CAD. The power series approach of [3] is strengthened in section three by describing how to automatically produce differential equations satisfied by the input function. It also shows how one can use the method recursively to handle multivariate problems, thereby confirming the possibility raised in [3]. Finally we present some more multi-valued simplification formulae.

2. CALCULATION OF BRANCH CUTS

Here we propose to represent the branch cuts using semi-algebraic sets. Following [5] if we restrict the functions considered to the case where we only allow nesting of square roots inside other functions and bar inverse functions from appearing in the denominators the set of branch cuts can be described by semi-algebraic sets. As such, they are now within the remit of CAD and an example was worked in [2] with the cuts being produced by hand before being given to a CAD engine. Here we now show how to automate that proceedure.

2.1 Rational Functions

Suppose firstly that we are working with one complex variable and that we do not allow nesting of roots inside elementary functions. We start by considering functions of the form $f(g(z))$ where f is an elementary inverse function, and $g \in \mathbf{C}(z)$, with $g(z) \neq 0$. The output should be a semi-algebraic set where the polynomials involved contain only 2 real variables.

We first make some observations of facts which make it possible to achieve this aim. Firstly, all the branch cuts for the square root, logarithm, and the 12 inverse (trigonometric, hyperbolic) functions have cuts which can be described in terms of sets of inequalites involving their real and imaginary parts s and t separately where $s, t \in \mathbf{R}$; that is, disjunctions of sets of the form $\{(s\, pred_1\, c_1), (t\, pred_2\, c_2)\}$ where $pred_i \in \{<, \leq, \geq, >\}$ and $c_i \in \{0, \pm 1\}$. For example, the cuts for $\mathrm{arcsinh}(z)$ are $\{(s = 0) \wedge (t \geq 1)\} \vee \{(s = 0) \wedge (t \leq 1)\}$. Further, for each cut of every function, we always have either $s = 0$ or $t = 0$. Re-writing g if necessary as $g(z) = p(z)/q(z)$, where $p, q \in \mathbf{C}[z]$, we obtain by the substitution $z \rightarrow x + yi$ and multiplying by the expression $\frac{\Re(q) - \Im(q)i}{\Re(q) - \Im(q)i}$ the equation:

$$g(z) = \frac{\Re(p)\Re(q) + \Im(p)\Im(q) + (\Re(q)\Im(p) - \Re(p)\Im(q))i}{\Re(q)^2 + \Im(q)^2}. \tag{1}$$

Then, for each separate cut of f, one sets up a system of the following form:

$$\begin{aligned} g(z) &= s + ti && (2)\\ s\ &pred_1\ c_1 && (3)\\ t\ &pred_2\ c_2. && (4) \end{aligned}$$

The choice of values for $pred_i$ and the c_i depends on f, and here we follow [10], although the method would be the same for any other choice of branch cuts. Without loss of generality, suppose that the input function f requires that $s = 0$. Then combining (2) and (3) we now have

$$\begin{aligned} g(z) &= ti && (5)\\ t\ &pred_2\ c_2. \end{aligned}$$

Now using (1), and equating real and imaginary parts in (5) we obtain

$$\begin{aligned} \Re(q)\Im(p) - \Re(p)\Im(q)\ &pred'\ t(\Re(q)^2 + \Im(q)^2) && (6)\\ \Re(p)\Re(q) + \Im(p)\Im(q) &= 0\\ t\ &pred_2\ c_2. \end{aligned}$$

Finally, since the denominator of (1) is always strictly positive, and we know c_2, (recall $c_2 = 1$ or $c_2 = -1$), we can determine $pred'$ and substitute for the relevant c_2 in (6) to get

$$\begin{aligned} \Re(q)\Im(p) - \Re(p)\Im(q)\ &pred'\ \pm(\Re(q)^2 + \Im(q)^2) && (7)\\ \Re(p)\Re(q) + \Im(p)\Im(q) &= 0. && (8) \end{aligned}$$

The case where we require that $t = 0$ is similar. Note that we have 2 equations in 2 real variables.

We can generalise this to the case where we have $h(f_i(g_i(z)))$ with h a rational function in the $f_i(g_i(z))$, i in $1 \ldots n$ say, and where we also allow the taking of complex conjugates. Then we apply the proceedure above to each $f_i(g_i(z))$ and then take the union; the only case which requires slight modification is when a $f_i(g_i(z))$ is of the form $f_i(\overline{g_i(z)})$. Here equation (5) (for example) becomes $g_i(z) = -ti$ using the fact that $\forall(z)\forall(w)(\overline{z} = \overline{w} \Leftrightarrow z = w), z, w \in \mathbf{C}$, and that $g_i(z)$ has no branch cuts. Finally we note that the whole proceedure clearly extends to an arbitrary number of complex variables.

2.2 Example

Suppose that we wish to calculate the branch cuts of $f(z)$, where

$$f(z) = \text{arcsinh}\left(\sqrt{\frac{z^2+z-5}{z^2+1}}\right).$$

For brevity, we skip some of the steps. The cuts for $\text{arcsinh}(z)$ were given above, and we illustrate the case for the cut given by $\{(s=0) \wedge (t<-1)\}$ with the other cut requiring a similar calculation. We obtain after substituting $s=0$ into (2) and taking the conjugate of its righ-hand side, the pair

$$\begin{aligned} \frac{z^2+z-5}{z^2+1} &= -ti \\ t &< -1. \end{aligned}$$

Now equation (6) becomes

$$\begin{aligned} (x^2-y^2+1)(2xy+2y) - (x^2-y^2+2x-5)(2xy) \quad pred' \\ -t((x^2-y^2+1)^2+(2xy)^2). \end{aligned}$$

Since we have $t<-1$, clearly $pred'$ is $>$. Finally, we obtain

$$\begin{aligned} (x^2-y^2+1)(2xy+2y) - (x^2-y^2+2x-5)(2xy) \\ > (x^2-y^2+1)^2+(2xy)^2, \\ (x^2-y^2+2x-5)(x^2-y^2+1) + (2xy+2y)(2xy) = 0, \end{aligned}$$

which can be of course be expanded and rewritten in the neater form of $\{p_1 > 0, p_2 = 0\}$ for the appropriate polynomials p_i.

2.3 Nesting of Square Roots

Suppose again that we have $f(g(z))$ with f elementary, but this time allowing $g(z)$ to contain square roots of arbitrary depth, although it will be seen that the arbitrariness of depth causes no further theoretical difficulties. On the other hand, this problem seems to be considerably harder than the non-nested case, and as such is work in progress. Despite this, a lot of examples are now possible, and we demonstrate the method in section (2.6). Regarding the input function $h(z) = f(g(z))$ as a expression tree, one starts to build the set of branch cuts for h by calculating the set of cuts for each square root, and taking the union. One could work, for convenience, from the leaves of the tree upwards, although this is not strictly necessary. Finally we calculate the set of cuts for f at the top-most level and adjoin these to the set we have built so far. It is sufficient then, to describe how to calculate this latter set of cuts for f, for any of the square roots can be handled in the same way, apart from those cases where the inputs to the roots only contain elements of $\mathbf{C}(z)$; this case has been tackled already in section (2.1). Essentially, one sets up an appropriate system again corresponding to equations (2),(3), and (4), and in order to obtain a polynomial system the problem is clearly that of removing the square roots from (2).

2.3.1 Removing the Square Roots

Suppose that without loss of generality we have the following system

$$\begin{aligned} g(z) &= s \qquad (9) \\ s &< 0 \end{aligned}$$

where $g(z)$ is of the form, $g(z) = \frac{c_1\sqrt{\cdots\sqrt{}}}{c_2\sqrt{\cdots\sqrt{}}} + \frac{c_3\sqrt{\cdots\sqrt{}}}{c_4\sqrt{\cdots\sqrt{}}}$, where we allow any expressions within the restrictions of [5] under the square roots, and with the $c_i \in \mathbf{C}[z]$. Then one can isolate one of the summands on the LHS of (9) by rearranging, and then square both sides of the equation to remove one of the terms involving roots. If one of the terms of $g(z)$ was just a polynomial say, then we are done. If not, repeating the process yields an expression without any roots, and now section 2.1 applies, ignoring for the moment the difficulties of (2.3.2). For more general $g(z)$ such as $\sqrt{p_1}+\sqrt{p_2}-\sqrt{p_3}$ the above proceedure will not terminate in general. An approach to this is as follows. Clearing denominators from (9) if necessary and rearranging we obtain an equation of the form $\phi(z,s)$ where $\phi(z,s)$ is a polynomial in $\sqrt{p_1}\ldots\sqrt{p_n}$ for some n. For each $\sqrt{p_i}$ we make the substituition $g_i = \sqrt{p_i}$. Then we can calculate the Gröbner basis for the set $\{{g_i}^2 - p_i, \phi(z,s)\}$ with respect to the variables $\{g_i, p_i \mid i = 1\ldots n\}$, and extract the polynomial in the p_i only.

Unfortunately squaring both sides of an equation creates an additional difficulty which we now describe.

2.3.2 Avoiding Spurious Branch Cuts

Suppose that we are given without loss of generality the system,

$$\begin{aligned} \sqrt{p(z)} &= f(s,z) \qquad (10) \\ s\, pred\, c \qquad (11) \end{aligned}$$

where f, p built from roots and polynomial functions as before. Now squaring (10) naively, we obtain $p(z) = f^2(s,z)$. But this equation represents the solution set of $\pm\sqrt{p(z)} = f(s,z)$, that is, for both branches of the square root, instead of just the principal branch. To avoid this, one only solves for p on the region where $csgn(f(s,z)) = 0, 1$, subject to the condition in (11). Therefore in order to produce an accurate set of cuts for the function we must in general add some polynomial constraints to the system; if we do not do this, then we can get results which are incorrect. For example, trying to find the cuts of $\text{arccosh}(\sqrt{z})$ for example would lead us to consider, after reduction, the system,

$$\begin{aligned} \sqrt{z} &= s \qquad (12) \\ s &< 1. \qquad (13) \end{aligned}$$

After naively squaring (14) and taking real and imaginary parts we obtain

$$\begin{aligned} x &= s^2 \qquad (14) \\ y &= 0 \qquad (15) \\ s &< 1, \qquad (16) \end{aligned}$$

from which we conclude incorrectly[3] that the branch cuts are the set $\{(x>0) \wedge (y=0)\}$ when we want
the set $\{(0 \le x < 1) \wedge (y=0)\}$ only.

More precisely, what we should do is to consider the following pair of systems:

[3] As in footnote 2, one can easily extend the deduction of $pred'$ corresponding to equation (7) by using a Maple function such as max or min on the region of consideration.

$$\begin{aligned} p(z) &= f^2(s,z) \\ \Re(f(s,z)) &> 0 \\ s\ pred\ c \end{aligned} \tag{17}$$

and

$$\begin{aligned} p(z) &= f^2(s,z) \\ \Re(f(s,z)) &= 0 \qquad (18)\\ \Im(f(s,z)) &\geq 0 \qquad (19)\\ s\ pred\ c, \end{aligned}$$

and we must convert equations (17),(18) and (19) into polynomial inequalities. In the case where $f^2(s,z)$ does not contain square roots, this reduces to the case of section 2.1. One slight difference that must be noted though is that here we may no longer be able to eliminate s from the system after equating real and imaginary parts in the same manner as 2.1.[4] In this case, we have to compute a three dimensional CAD in the variables s, x, y as opposed to a CAD in the original variables of the problem, which is clearly a drawback in terms of complexity. Also the difficult issue of providing the user with information on where the identity holds[21] is exacerbated. If $f^2(s,z)$ does contain square roots, then we must recursively apply this proceedure. We note that this may require us to add new variables. For example, converting the system given by (18), (19) and (20) would require us to set up the system

$$\begin{aligned} p(z) &= f^2(s,z) \qquad (20)\\ f(s,z)) &= s_1 \qquad (21)\\ s_1 &> 0 \qquad (22)\\ s\ pred\ c \end{aligned}$$

Note that we must retain the original set of constraints on s. The case where we have to use Gröbner bases to remove the square roots is problematic as it does not tell us how explicitly which equations have been squared up into generate the appropriate side-conditions as in (17), (18) and (19) and consequently requires further investigation.

This generalises exactly the same as before to the case where we have $h(f_i(g_i(z)))$ with h a rational function in the $f_i(g_i(z))$, i in $1 \ldots n$.

2.3.3 Example

Suppose that $h(z) = \sqrt{\sqrt{p_1} - \sqrt{p_2}}$ for some $p_1, p_2 \in \mathbf{C}[z]$. To determine the cuts for $h(z)$ we first calculate the cuts for $\sqrt{p_1}$ and $\sqrt{p_2}$ by applying section (2.1). After reduction and rearranging, we must consider the pair of systems:

$$\begin{aligned} \sqrt{p_1} &= s + \sqrt{p_2} \qquad (23)\\ s &< 0 \\ \Re(s + \sqrt{p_2}) &> 0, \qquad (24) \end{aligned}$$

[4] In some cases this can be done, but we cannot make this algorithmic currently.

and

$$\begin{aligned} \sqrt{p_1} &= s + \sqrt{p_2} \qquad (25)\\ s &< 0 \\ \Re(s + \sqrt{p_2}) &= 0, \qquad (26)\\ \Im(s + \sqrt{p_2}) &\geq 0, \qquad (27) \end{aligned}$$

where we have the additional side-conditions of (24), (26) and (27) in order to square up the equations (23) and (25) without including the negative branch in the solution set of the system. We show how to convert (24) into a set of polynomial inequalities, with the remaining two cases being similar. So we set up the following system, once again using the fact that the imaginary component $t = 0$, to obtain

$$\begin{aligned} s + \sqrt{p_2} &= s_1 \qquad (28)\\ s_1 &> 0 \qquad (29)\\ s &< 0, \\ & \qquad (30) \end{aligned}$$

where we have had to add the real variable s_1. Again we must rearrange (28) to obtain

$$\sqrt{p_2} = s_1 - s, \tag{31}$$

in order eliminate the square root in (31) but this time, the recursion stops here since the right-hand side contains no square roots; hence we obtain the pair

$$\begin{aligned} p_2 &= {s_1}^2 - 2s_1 s + s^2 \qquad (32)\\ s_1 - s &> 0, \qquad (33) \end{aligned}$$

where (33) is the side-condition required in order to square up (31) safely. Finally we can take the side-conditions of (32), (33) and (29) and replace equation (24) with them in that system, and now (safely) square up (23) to obtain the system,

$$\begin{aligned} p_1 &= s^2 + 2s\sqrt{p_2} + p_2 \qquad (34)\\ s &< 0 \\ p_2 &= {s_1}^2 - 2s_1 s + s^2 \\ s_1 - s &> 0 \\ s_1 &> 0. \\ & \qquad (35) \end{aligned}$$

2.4 Special cases

Suppose that we have $f(g(z))$ with f any elementary function. If $g(z) = c_1\sqrt{g_1(z)} + \ldots + c_n\sqrt{g_n(z)}$, where $g_i(z)$ are any functions allowed by our restrictions, with real constants $c_i \geq 0$ then $(\forall z)\, \mathrm{csgn}(g(z)) \neq -1$. Hence we do not need to consider the cuts for f which are wholly on the negative real or negative imaginary axis, when calculating the top-level cuts for $f(g(z))$. Those cuts which are partially on these axes can be restricted to the cuts, $\{z \mid (0 \leq \Re(z)\ pred\ 1) \wedge (\Im(z) = 0)\}$ or $\{z \mid (\Re(z) = 0) \wedge (0 \leq \Im(z)\ pred\ 1)\}$ as appropriate, with $pred$ being either of $\leq, <$. If we have $g(z) = \frac{c_1\sqrt{g_1(z)}}{c_2\sqrt{g_2(z)}}$ then we have that $\arg(g(z)) \in (-\pi, \pi)$. Hence we can ignore cuts of f that are on the negative real axis. Clearly using these where possible will save a great deal of computation time used by the steps in section (2.3), especially when they are several levels of nested roots, and

cover sufficiently many cases to consider them worth putting in an implementation. They clearly hold for any number of complex variables.

2.5 Remark

One problem that has not yet been tackled is the detection of *removable* branch cuts. An example is the function $\log(z+1) - \log(z-1)$ which has the removable cut $\{(x \leq -1) \wedge (y = 0)\}$, although the proceedure in section 2 would not detect this. We note that [14] was able to eliminate some of these, although again, only for special cases. Here we note a fact, not explicitly mentioned in [14] or previous papers by the authors. If one ignores the removable cuts, then when one is testing a region that contains a removable cut, if the sample point we choose happens to be within the removable cut, one would conclude should the identity hold at this point that it is true on the whole of the region; should the identity be false, one would conclude that is false on the entire region. Of course, both these conclusions are false in general. Note also that this can only occur on the cells of less than full dimension. Unlike [14] however, we use CAD to generate the sample points, and the cells corresponding to the removable cuts are produced automatically. This is because removable cuts only occur where the polynomials have common zeros; the sample point test will then reveal whether such a cell actually corresponds to a removable cut or not.[5]

2.6 Sample-Point Testing

Clearly the complexity of this section will be depend on the number of cells produced in section two. Once again the use of PCAD will reduce this. The remarks made in [5, 2] still apply.

3. USE OF POWER SERIES

The first step of [3] requires us to find an algebraic differential equation satisfied by h where $h = f - g$. This is fine if we happen to be starting from a DE, but is less convenient if we have an expression such as $\sqrt{x^2}$ to work with. In this section we show how to compute DEs for expressions by the use of Gröbner bases.

Suppose, first, we have DEs for f and g of the form

$$f^{(n)} + L(f) = 0 \quad (36)$$
$$g^{(m)} + M(g) = 0 \quad (37)$$

where L and M are differential operators of orders less than n and m respectively. What is a DE for $h = f - g$? Now, by successive differentiation, and substitutions $f^{(n)} = -L(f)$ and $g^{(m)} = -M(g)$ where appropriate we have

$$\begin{aligned} h - (f - g) &= 0 \\ h' - (f' - g') &= 0 \\ h'' - (f'' - g'') &= 0 \\ &\cdots \\ h^{(r)} - (L_r(f) - M_r(g)) &= 0 \end{aligned}$$

for some differential operators of L_r and M_r of orders less than n and m respectively.

[5] subject to the problems of section (2.9).

Now regard the above as a system of equations in variables h, f, g, h', f', g', h'', f'', g'', and so on, and eliminate the fs and gs using Gröbner bases. This leaves us with a DE for h.

Generalising, a similar process will compute a DE for $h = p(f, g)$ where p is some polynomial.

Another case is where g is defined in terms of f and we want to find a DE for g:

$$\begin{aligned} f^{(n)} + L(f) &= 0 \\ g^{(m)} + M(f, g) &= 0 \end{aligned}$$

We now have

$$\begin{aligned} g^{(m)} + M(f, g) &= 0 \\ g^{(m+1)} + M(f, g)' &= 0 \\ &\cdots \end{aligned}$$

and by reducing by $f^{(n)} = -L(f)$ where necessary we can determine a system of equations for $g^{(i)}$ in the $f^{(j)}$ which can be solved to produce a DE for g.

In general, we can store the DEs for a few simple expressions (e.g., $g = f^n$ satisfies $g' - ngf'/f = 0$; $g = \log f$ satisfies $g' - f'/f = 0$) and use the above to combine them into more complicated expresions.

3.1 Example

$g = \sqrt{x^2} = (x^2)^{1/2}$. One of the many ways of finding a DE for g is as follows. Set $f = x^2$, so $g = f^{1/2}$. Now

$$\begin{aligned} f' - 2x &= 0 \\ g' - gf'/2f &= 0 \end{aligned}$$

which reduces to:

$$g' - xg/f = 0$$

Differentiating:

$$g'' + xg/f - g/f = 0$$

This gives us equations

$$\begin{aligned} fg'' + xg - g &= 0 \\ fg' - xg &= 0 \end{aligned}$$

in variables f, g, g' and g''. Eliminating f, we find

$$xgg'' + xg^2 - gg' = 0,$$

which is more complicated than we might first have expected. It has solutions $g = \pm\sqrt{ax^2 + b}$ for constants a and b. We note in passing that g also satisfies the simpler $gg' - x = 0$.

In general, this procedure will find a DE, but it may well not be of the smallest possible order. For example, $f = x^2$, $g = f^3$ will generate a DE of order 2, not the order 1 DE for $g = x^6$ we might hope for. As we saw in [3] this has a direct impact on the number of terms of the power series one has to check are zero. Another difficulty when combining several expressions such as $f - g$ is that the individual DE's produced from f and g may not be in the form of (13) and

(14). One approach would be to simply add such DEs as they stand to the Gröbner basis, although this will generally increase computation time.

We should note that we are using polynomial Gröbner bases, and not differential Gröbner bases[7, 19] so we avoid the termination problems that they have.

3.2 Several Variables

In [3] we suggested the possibility of extending the method of [25] to the case of several complex variables. Here we demonstrate that this can be done by giving an example, applying the method *recursively*, as opposed to generalising the actual method of [25].

Example

Consider

$$h(z) \stackrel{?}{=} \arctan\left(\frac{x+y}{1-xy}\right) - \arctan(x) - \arctan(y). \quad (38)$$

However, the method of [3], based on [25], is only directly applicable to univariate equations. We therefore write $K = \mathbf{C}(x)(elem)$, the elementary function closure of $\mathbf{C}(x)$, and work in $R = K[[y]]$, with our derivation being $\frac{\mathrm{d}}{\mathrm{d}y}$. This may cause us to ask question in K, which we will map into $\mathbf{C}[[x]]$, and solve recursively. However, rather than use the $'$ notation, we will (to avoid confusion between $\mathbf{C}[[x]]$ and $K[[y]]$) use the standard PDE notation and write h_y for $\frac{\mathrm{d}h}{\mathrm{d}y}$. We recall from [25, 3] that δ is the diferential operator $y\frac{\partial}{\partial y}$. $F^{(r)}$ will denote $\delta^r F$.

In this context, the key lemma of [25] becomes the following result.

LEMMA 1. *[25] After a transformation of the form $h \to h_0 + h_1 y + \cdots h_v y^v + \hat{h} y^{v+1}$, and division by a suitable power of y, we may assume that*

$$Q = LH + yM, \quad (39)$$

where $L \in K[\delta]$ is non-trivial, and $M \in R\{H\}$, which is the differential polynomial extension of R by H and its derivatives.

First we must determine Q, the differential equation satisfied by h and its formal analogue H.

$$\begin{aligned} h_y &= \frac{1-xy+(x+y)x}{(1-xy)^2}\frac{1}{1+\left(\frac{x+y}{1-xy}\right)^2} - \frac{1}{1+y^2} \\ &= \frac{1+x^2}{(1-xy)^2+(x+y)^2} - \frac{1}{1+y^2} \\ &= \frac{1+x^2}{(1+x^2)(1+y^2)} - \frac{1}{1+y^2} \\ &= 0 \end{aligned}$$

By symmetry, the equivalent $h_x = 0$ as well, which means that, as observed in [3] in the univariate case, h is locally a constant, so it suffices to evaluate it at a sample point in each region of the decomposition of $\mathbf{C}^2$ induced by the *potential*[6] branch cuts of h. (i.e. the union of the branch cuts of f and g) However, to show the power of the multivariate extension of the method of [25], we will ignore this short cut. The branch cuts may be readily found by the method of section (2.1) to be the variety $xy - 1 = 0$; given which QEPCAD would produce 5 regions. As an example we demonstrate the method for the regions R_1 and R_3 with sample points (0,0) and (-1,-1) respectively. subsectionThe region R_1 Here there is no need to translate the differential equation $h_y = 0$ so Q is $H_y = 0$, which we can multiply by y to get $yH_y = \delta H = H^{(1)} = 0$.

(c-1.5) $\frac{\partial Q}{\partial H^{(1)}} = 1$, so $v = 0$. Proceeding formally, $h(0) = \arctan\left(\frac{x+0}{1-0x}\right) - \arctan(x) - \arctan(0) = 0$, so we write $h = y\hat{h}$, and then $Q = y\hat{h}_y + \hat{h}$. In the formalism of equation (39), this is $H^{(1)} + H^{(0)}$, or $(\delta+1)\hat{H}$.

(c-1.6) $\Lambda = k + 1$, so $s = -1$. THere is therefore nothing to check.

Hence the equation is true on R_1.

3.3 The region R_3

Our sample point is now $(x = -1, y = -1)$. Here again, x does not "go away" so neatly, and we are forced to recurse in the number of (analytic) variables: in this case x and y.

3.3.1 *Initial exploration in y*

(c-3.5) The transformation $y \to y + 1$ does not change Q, and again $\frac{\partial Q}{\partial H^{(1)}} = 1$, so $v = 0$. Proceeding formally, $h(0) = \arctan\left(\frac{x-1}{1+1x}\right) - \arctan(x) - \arctan(-1)$. This has to be treated as a sub-problem in x.

3.3.2 *The subproblem in x*

In this section we re-use the notation of the general algorithm, so that h, H, Q etc. in this subsection have nothing to do with the previous work.

$$h = \arctan\left(\frac{x-1}{1+x}\right) - \arctan(x) - \arctan(-1), \quad (40)$$

and

$$\begin{aligned} h_x &= \frac{-1(1+x)-1(x-1)}{(1+x)^2}\frac{1}{1+\left(\frac{x-1}{x+1}\right)^2} - \frac{1}{1+x^2} \\ &= \frac{-x-1}{1+x^2}. \end{aligned}$$

so Q is $H_x + \frac{x+1}{1+x^2} = 0$. We can observe at this point that it is impossible for $h = 0$ to satisfy this equation, but nevertheless let us continue algorithmically.

(c-3-1.5) We will now transform our independent variable to $z = x + 1$, which makes Q into $H_z + \frac{z}{1+(z-1)^2}$, which we then multiply by z. In the formalism of equation (39), this is $H^{(1)} + \frac{z^2}{2-2z+z^2}$.

(c-3-1.6) $\Lambda = k$, so $s = 0$.

(c-3-1.7) We must therefore check that $h_0 = h|_{z=0}$ is zero. In terms of the original untransformed variables, this is

[6] Of course, if the identity was correct, $h = 0$ and has no branch cuts.

$h|_{x=-1}$, which is $\arctan\left(\frac{-1-1}{1-(-1)(-1)}\right) - \arctan(-1) - \arctan(-1)$, which, according to the branch cuts in [10], evaluates to $\frac{\pi}{2} + \frac{\pi}{4} + \frac{\pi}{4} = \pi$.

Hence h (in terms of x) is a series of the form $\pi + h_1(x+1) + \cdots$

(c-3.5) continued In the previous section, we established that, in the notation of this section, $0 \neq h(-1) = h|_{y=-1} \in K[[y]]$. We will now transform our independent variable to $z = y - 1$, so that Q is now $H_z = 0$. Multiplying by z, we get $zH_z = \delta H = 0$.

(c-3.6) $\Lambda = k$, so $s = 0$.

(c-3.7) We must check that $h_0 = h|_{z=0} = h_{y=-1} = 0$, where $h_{y=-1}$ brings is back to the untransformed variables. This is $\arctan\left(\frac{x-1}{1+x}\right) - \arctan(x) - \arctan(-1)$. However, this is the right-hand side of equation (40) and, as shown in section 3.3.2, is non-zero.

Hence the equation is false on R_3, and indeed, since we know the discrepancy is constant, we know that it is π.

4. IMPLEMENTATION

For the decomposition step we have harnessed the QEPCAD system[22]. The fact that it allows boolean connectives always allows us to fomulate the problems with faithfully from the systems we generate in section (2); each equation in a system are to be conjoined using a $\wedge$ and we conjoin systems by using the $\vee$ connective. We were able to extract the sample points and their corresponding dimensions from the PCAD data structure as input to our master Maple program. A Prototype of section (2.1) has been implemented, and on the limited testing we have done, returns most examples within a second or two.

Regarding the use of `simplify(...,symbolic)`[7]

- it will in general apply several transformations to obtain g.
- We could ensure this step terminates by putting 'sensible' time limits on the computation, although so far all examples we have tried have returned a result (simplified or not) within a second or two.
- In some cases, human intervention may be necessary to obtain a simplification that would, to most observers, seem simpler than the starting function. For example Maple 8 decides that $\sqrt{x}\sqrt{y}$ is a simplification of $\sqrt{xy}$ and not vice-versa; it is unclear why this is so.

The problem with this is that a chain of elementary simplifications may ultimately give a correct simplification, as branch cuts may effectively cancel each other out.[8] Thus we may erroneously reject the proposed simplification g at the first step if we apply a transformation that is not correct in the single-valued sense. If we do have $f = g$, then clearly the first approach would have been more efficient. At this stage we are mainly investigating the general feasibility of the method, so regarding computational speed, we feel that explicit timings are not appropriate here; see the remarks in section (5.2).

4.1 Remark

We note that one can use the `solve` command in Maple as in optimisation for step 1. Given a system of polynomial inequalities, if Maple returns one or more conditions on each of the variables of the form $varpredc$ then we can convert this into a form suitable for QEPCAD input. In general the package will then produce fewer cells than the original formula, since the only polynomials occuring have degree 1.

4.2 Complexity

It is known that CAD has doubly exponential complexity in the number of variables. To give an estimate of complexity for the polyalgorithm is a difficult problem; this is largely due to fact that there are several different algorithms involved, and some of these have not yet been quantified- in [25] no bounds for complexity were given, although Van der Hoeven believes it should be possible, and nor for PCAD whilst consideration of the method in [9] shows that it is always as least as efficient as CAD, and the examples given there surely convince one that it is often much more efficient. Additionally, the inevitable use of Maple commands like `solve` and `simplify(...,symbolic)` as a black box causes further difficulty. Hence the best approach may just on experience; fortunately in this area, examples are plentiful and easy to manufacture!

5. CONCLUSION & FUTURE DIRECTIONS

We have been able, by combining the methods of [5, 3] to simplify a large class of examples. The fact that both approaches require CAD is clearly a limitation though, as some simple examples such as $\log(\exp(z)) - z$ do not have cuts which are semi-algebraic. One might be able to use the method of Pfaffian decomposition[16], although the feasibility of the approach is not yet clear. One cannot produce an optimal number of cells in general using the techniques of partial CAD, although powerful, without applying a method such as [6] after the initial decomposition had been carried out, and an alternative may be to adapt the method of [15], which finds a point in every connected component of the plane in sub-exponential time, to distinguish between sets of measure zero which overlap. Nevertheless, we feel that the use of CAD forms a useful testbed for the method, and is currently one of the more widely available approaches in CAS.

6. SOME MORE MULTIVALUED IDENTITIES

In table one we present some more multi-valued simplification rules. More identities will be found from the website, `http://www.cs.bath.ac.uk/~cspnp/simplification/mvfunc.html`[10]

[7] Note that this paper does not address the fundamental question of what we actually mean by 'simpler', see [20] for further discussion.

[8] Actually if we could eliminate all *removable* cuts, then this won't happen; this is covered in section

[10] which will be continuously updated.

Rule	Single-valued counter-example
$\exp(\mathrm{Log}(x)) = \{x\}$	no counterexample
$(f \circ f^{-1})(x) = \{x\}$	in general[9]
$\mathrm{Log}(\exp(x)) = \{x + 2n\pi i \mid n \in \mathbf{Z}\}$	$x = 2\pi i$
$\mathrm{Log}(z^n) \supset n\,\mathrm{Log}(z), n \in \mathbf{Z}$	$(eg.\, n = 3)\; z = -i$
$\cos(\mathrm{Arccos}(x)) = \{x\}$	no counter-example
$\mathrm{Arccos}(\cos(x)) =$	$x = \frac{3\pi}{2}$
$\{x + 2n\pi \mid n \in \mathbf{Z}\} \cup$	
$\{-x + 2n\pi \mid n \in \mathbf{Z}\}$	
$\mathrm{Arcsin}(z) \cup \mathrm{Arcsin}(-z) =$	$\mathrm{Arctan}\left(\frac{z}{\sqrt{1-z^2}}\right)$
$\mathrm{Arcsinh}(\sinh(x)) =$	$x = \pi$
$\{x + 2n\pi i \mid n \in \mathbf{Z}\} \cup$	
$\{-x + (2n+1)\pi i \mid n \in \mathbf{Z}\}$	
Conjugated formulae:	
$\overline{\mathrm{Log}(x)} = \mathrm{Log}(\overline{x})$	$x = -1$
$\overline{\mathrm{Arctan}(x)} = \mathrm{Arctan}(\overline{x})$	$x = 2I$
$\overline{\mathrm{Arctanh}(x)} = \mathrm{Arctanh}(\overline{x})$	$x = -2$

Table 1: Multi-valued simplification formulae

7. REFERENCES

[1] Abramowitz, M., and Stegun, I. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. *US Government Printing Office* (1964; 10th printing, 1972).

[2] Beaumont, J., Bradford, R., and Davenport, J. Towards Better Simplification of Elementary Functions. *Pre-print, University of Bath, England.* (2002). `http://www.cs.bath.ac.uk/~cspnp/simplification/mvfunc.html`

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

[4] Bradford, R., Corless, R., Davenport, J., Jeffrey, D., and Watt, S. Reasoning about the Elementary Functions of Compelx Analysis. *Ann. Mathematics and Artificial Intelligence 36*, (2002), 303–318.

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

[6] Brown, C. W., Simple CAD Construction and its Applications. *J. Symbolic Computation*, (31):521–547, 2001.

[7] Carrà Ferro, G. Gröbner Bases and Differential Algebra. *Proc. AAECC-5 (ed. L. Huguet & A. Poli), Springer Lecture Notes in Computer Science 350, Springer-Verlag*, (1987), pp. 129-140.

[8] Collins, G. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages* (1975), vol. 33 of *Springer Lecture Notes in Computer Science*, Springer-Verlag, pp. 134–183.

[9] G. E. Collins and H. Hong, Partial Cylindrical Algebraic Decomposition for Quantifier Elimination. *J. Symbolic Computation*, 12(3):299–328, 1991.

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

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

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

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

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

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

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

[17] Henrici, P. Applied and Computational Complex Analysis. Vol.1, Wiley and Sons, (1974).

[18] Kahan, W. Branch Cuts for Complex Elementary Functions. *The State of Art in Numerical Analysis* (1987), 165–211.

[19] Mansfield, E. L. Differential Gröbner Bases. Ph.D. Thesis, University of Sydney, 1992.

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

[21] Patton, C. M. A Representation of Branch-Cut Information *SIGSAM Bulletin* **116**(1996), pp. 21-24.

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

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

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

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