User Interface Design for Geometrical Decomposition Algorithms in Maple

Changbo Chen[1], James H. Davenport[2], John May[3], Marc Moreno Maza[1], Bican Xia[4], Rong Xiao[4], and Yuzhen Xie[5]

[1] Department of Computer Science, University of Western Ontario
changbo.chen@gmail.com,moreno@csd.uwo.ca
[2] Department of Computer Science, University of Bath
J.H.Davenport@bath.ac.uk
[3] Maplesoft jmay@maplesoft.com
[4] School of Mathematical Sciences, Peking University
{xbc@math.pku.edu.cn,akelux@gmail.com}
[5] CSAIL, Massachussets Institute of Technology, Cambridge MA, USA
yxie@csail.mit.edu

Abstract. As computer algebra develops, it handles more sophisticated objects, many of which have no precise parallel in conventional mathematics, since mathematicians have handled the concepts on an ad hoc basis. Furthermore, by definition, computer algebra must handle these objects algorithmically, and present them to the user. This is particularly a challenge when the user may not be intimately familiar with the object, and all the special cases that may occur.
We present various issues connected with this in the context of equation solving, and show how the 'piecewise' construct of Maple [3] can be employed to build representations of solution objects that:

1. Are intuitive in simple cases;
2. Use familiar base constructs;
3. Allow 'delayed evaluation' of difficult special cases, which the user may not actually be interested in.

1 Introduction

Algorithms decomposing geometrical entities into meaningful components challenge the designers of computer algebra systems for a variety of reasons.

(i) The user's intuition is honed on simple examples, e.g. the solution to the equations is a line, often with rational coefficients. The user may never have seen the full generality of such decompositions. However, we should not invent an entirely new language, but should rely on techniques that the user *does* understand.
(ii) The output of these algorithms may consist of components of different mathematical nature (points, curves, surfaces, etc.). Such a description is usually given largely in words, e.g. (for Whitney's umbrella)

"the solutions to $x^2 = y^2 z$ are the parametric plane $x = uv$; $y = u$; $z = v^2$ together with the negative z-axis."

Providing type information and structured results is certainly important for the expert user in order to analyze the output. On the other hand, over-structured results may discourage the non-expert user.

(iii) The algebraic expressions encoding the output components may be huge and the user will almost certainly not wish to see them in the first place, if at all. Even with more straight-forward algorithms, computer algebra systems are guilty of showing the user the trees, rather than the wood. For example, since the usual definition of 'Gröbner base' is "a set of polynomials such that ...", computer algebra system designers insist on showing the user the whole polynomials, rather than just the leading monomials.

(iv) Even worse, some of these objects may be totally useless to the user as they correspond to degenerate cases that are not of practical interest. The calculation of those degenerate cases may also be dramatically more expensive. However, the algorithm has no way of knowing which, if any, the user will want, and the algorithm designer will mislead the user if it doesn't show at least the *existence* of these solutions. We therefore use Maple's facility[6] to hold computations "inert" until the user explicitly asks for their value in order to represent the existence of these degenerate cases.

In this paper, we consider three kinds of such decomposition algorithms, with different challenges regarding their user interface in a computer algebra system. The first algorithm, studied in Sections 2, solves systems of polynomial equations and inequations, with parameters. Because it is based on the algebraic concept of *triangular decomposition* and inspects all possible behavior of the solutions depending on the parameters, the output of this algorithm is called *Comprehensive Triangular Decompositions* (CTD) of the input system.

On an input set of polynomials in n variables, the second algorithm, see Sections 3, decomposes the real space $\mathbf{R}^n$ into cells such that the sign of each input polynomial does not change in each cell. Because of the shape of those cells, this second algorithm is called *Cylindrical Algebraic Decomposition* (CAD).

The third algorithm, discussed in Section 5 and called *Real Triangular Decomposition* (RTD) takes as input a system of polynomial equations, inequations and inequalities. The output is based again on the algebraic concept of triangular decomposition. In contrast to the CTD, no parameters are involved and the purpose is to determine which solution components have points with real coordinates. Both RTD and CAD face additional challenges due to the inherent difficulties of manipulating *parametric* real (algebraic) numbers exactly: see Section 4.

To a first approximation, each of these three algorithms produces a list of components such that the following two conditions hold:

1. each solution of the input system belongs to at least one component, and

[6] `http://www.maplesoft.com/support/help/view.aspx?path=file06209#value` describes this facility.

2. each point in each component is a solution of the input system.

By its "comprehensive nature", a CTD is in fact a family of decompositions of the input system rather than a single decomposition; each of these decompositions is associated to a case (or constructible set) in the parameter space. It is therefore desirable to organize the equations and inequations encoding those cases in a synthetic way such that for a given parameter value the user can easily access the corresponding (specialized) decomposition. We show in this paper how the 'piecewise' construct of Maple helps achieving this goal.

In the case of RTD, we propose a "lazy evaluation" model, where only the "main components" are evaluated in the first place. The calculations of the other components are performed only if the user wants too. We rely again on the `piecewise` construct of Maple together with the existing features of this system for delaying evaluations. While delayed evaluation is not new in computer algebra. it has generally been used for infinite objects such as Taylor series [9], where it is a necessity. Here, we are using it for two reasons:

- the *cost* of computing the special cases may greatly outweigh the cost of computing the generic case;
- the *size* of the special cases may drown the generic case.

Between the CTD and the RTD cases, we apply similar ideas to CAD, since this algorithm supports applications where a complete decomposition is required in the first place.

These three geometrical algorithms decompose systems of polynomial equations, inequations and inequalities. The components that they compute have a more complex structure (or representation) than those occurring in algebraic decomposition algorithms, such as primary decomposition of polynomial ideals. In this latter case a component is simply given by a system of generators (generally a Gröbner basis) and no inequations and inequalities are involved.

The decomposition algorithms implemented by the computer algebra systems such as CoCoA, MacCaulay, Magma, Singular are only primary decomposition and its variants (prime decomposition, equidimensional decomposition, triangular decomposition in dimension zero). Several implementations of CAD are available in dedicated C libraries (QEPCAD) or in the computer algebra systems Mathematica and Reduce proposing interfaces specific to CAD and which do not adapt to CTD or RTD. In our work, there is a desire of standardizing those different geometrical decompositions, as much as possible.

2 Comprehensive Triangular Decompositions

Consider the parametric polynomial system $\{vxy + ux^2 + x, uy^2 + x^2\}$, where x, y are unknowns and u, v are parameters. The näive user would probably try to handle this system in Maple with:

```
> solve({v*x*y+u*x^2+x, u*y^2+x^2},{x,y});
```

$$\left[\{x=0,y=0\},\left\{x=-\left(\frac{v\left(-v+u\sqrt{-u}\right)}{u^3+v^2}+1\right)u^{-1},y=\frac{-v+u\sqrt{-u}}{u^3+v^2}\right\},\right.$$
$$\left.\left\{x=-\left(-\frac{v\left(v+u\sqrt{-u}\right)}{u^3+v^2}+1\right)u^{-1},y=-\frac{v+u\sqrt{-u}}{u^3+v^2}\right\}\right]$$

This answer is useful, for some values of the parameters, but not for $u=-1, v=1$ for example. If the user will be evaluating the solution at many values of the parameters (not known before-hand), then an answer that specializes correctly at all values would be preferred.

In the current version of Maple, namely release 13, there are tools available to experts to compute the solutions for all possible parameter values. One way to do this with is with a Comprehensive Triangular Decomposition [4], which is done below. We signal the split between parameters and indeterminates with the argument '2', meaning that R is to be viewed with two variables (x and y) with the rest (u and v) being parameters (see also point C in the "further work" section).

```
> R := PolynomialRing([x,y,u,v]):
>  ctd := ComprehensiveTriangularize([v*x*y+u*x^2+x, u*y^2+x^2], 2, R);
ctd := [regular_chain, regular_chain, regular_chain, regular_chain], [
    [constructible_set, [1, 3]], [constructible_set, [1, 4]],
    [constructible_set, [1, 2]]]

> seq(Info(ctd[2][i][1], R), i=1..nops(ctd[2]));
                        2    3       2    3
              [[], [u, v  + u ]], [[v  + u ], [u]], [[u], [1]]

> map(Equations, ctd[1], R);
                                2  2        2    3   2
[[x, y], [x, u], [(v y + 1) x - u  y , 1 + (v  + u ) y  + 2 v y],
                   2  2             2    3
    [(v y + 1) x - u  y , 1 + 2 v y, v  + u ]]
```

Observe that the output is a structured object from which the expert can extract the necessary information, as shown by the last two commands: the first command extracts the cases (given by constructible sets) which form a partition of the u, v-parameter space, and the second one extracts the components solving x, y (by so-called regular chains). The way to read the output is then as follows: for each case, the corresponding components have their indices is the list next to the case.

For the work presented in this paper, we have implemented a new interface[7] to the command `ComprehensiveTriangularize` which produces the output below. The relation between each `constructible_set` and its associated

[7] Of course, we could have written a special-purpose printer to display the results shown above in a more hierarchical fashion, but one point of this paper is that *existing* tools can produce better output, if properly enlisted.

`regular_chain`'s is shown directly using a piecewise function in equation (1), which is produced with Maple's `latex` command (hand-edited for line breaks only).

$$\begin{cases} [\{x=0, y=0\}, \{(vy+1)\,x - u^2y^2 = 0, & \\ 1+\left(v^2+u^3\right)y^2 + 2\,vy = 0, & \\ v^2+u^3 \neq 0, vy+1 \neq 0\}] & \mathit{And}\left(u \neq 0, v^2+u^3 \neq 0\right) \\ [\{x=0, y=0\}, \{1+2\,vy = 0, v^2+u^3 = 0, & \\ (vy+1)\,x - u^2y^2 = 0, 2\,v \neq 0, vy+1 \neq 0\}] & \mathit{And}\left(v^2+u^3 = 0, u \neq 0\right) \\ [\{x=0, y=0\}, \{u=0, x=0\}] & u = 0 \end{cases} \tag{1}$$

This can be further processed to explicit solutions in the style of `solve`.

$$sol := \begin{cases} [\{x=0, y=0\}, & \\ \quad \{x = -\frac{u\left(-v^2+2\,vu\sqrt{-u}+u^3\right)}{v^3\sqrt{-u}+v\sqrt{-u}u^3+u^2v^2+u^5}, y = \frac{-v+u\sqrt{-u}}{v^2+u^3}\}, & \\ \quad \{x = -\frac{u\left(v^2+2\,vu\sqrt{-u}-u^3\right)}{v^3\sqrt{-u}+v\sqrt{-u}u^3-u^2v^2-u^5}, y = -\frac{v+u\sqrt{-u}}{v^2+u^3}\}] & u \neq 0 \text{ and } v^2+u^3 \neq 0 \\ [\{x=0, y=0\}, \{x = \frac{u^2}{2\,v^2}, y = \frac{-1}{2\,v}\}] & u \neq 0 \text{ and } v^2+u^3 = 0 \\ [\{x=0, y=y\}] & u = 0 \end{cases} \tag{2}$$

Above each of the three cases (or `constructible_set`s) on the right hand, the solutions of the input system are continuous multivalued functions of the parameters.

We note that, for a given parameter value, one could easily access the corresponding (specialized) solution by use of Maple command `eval` and the existing treatment of piecewise-defined functions [3]. So, with this structure, the user can easily specialize the solution correctly at any complex number:

```
> eval(sol, {u=1, v=1});
       {x = 0, y = 0}, {x = -1/2 - 1/2 I, y = -1/2 + 1/2 I},
       {x = -1/2 + 1/2 I, y = -1/2 - 1/2 I}
```

3 Cylindrical Algebraic Decomposition

The original paper [6, p. 149] defined a cylindrical algebraic decomposition as an indexed sequence of subsets, known as cells, of $\mathbf{R}^n$. In practice, of course, one has the formulae defining the sets, rather than the sets themselves, and it is normal to have a sample point in each cell. If we follow this definition, and represent the formulae by regular chains [5], we end up with a decomposition such as Figure 1, which is a decomposition of $\mathbf{R}^2$ induced by $xy - 1$.

While relatively compact, this is almost unreadable, even to one versed in the theory. Simply by extracting the cases, and constructing a corresponding expression with `piecewise`, we obtain Figure 2.

Note that we are no longer explicitly showing the cell indices, as they are implicit in the nested `piecewise` structure, e.g. the first row *has* to be `[1,1]`.

Fig. 1. Original CAD

```
[[[1, 1], [regular_chain, [[-1, -1], [-2, -2]]]], [[1, 2], [regular_chain,
[[-1, -1], [-1, -1]]]], [[1, 3], [regular_chain, [[-1, -1], [0, 0]]]], [[2,
1], [regular_chain, [[0, 0], [0, 0]]]], [[3, 1], [regular_chain, [[1, 1],
[0, 0]]]], [[3, 2], [regular_chain, [[1, 1], [1, 1]]]], [[3, 3],
[regular_chain, [[1, 1], [2, 2]]]]]
```

Fig. 2. Reformulated CAD: `latex(CylindricalAlgebraicDecompose(F, R))`

$$\begin{cases} \begin{cases} [\mathit{regular\_chain}, [[-1,-1],[-2,-2]]] & x < y^{-1} \\ [\mathit{regular\_chain}, [[-1,-1],[-1,-1]]] & x = y^{-1} \\ [\mathit{regular\_chain}, [[-1,-1],[0,0]]] & y^{-1} < x \end{cases} & y < 0 \\ [\mathit{regular\_chain}, [[0,0],[0,0]]] & y = 0 \\ \begin{cases} [\mathit{regular\_chain}, [[1,1],[0,0]]] & x < y^{-1} \\ [\mathit{regular\_chain}, [[1,1],[1,1]]] & x = y^{-1} \\ [\mathit{regular\_chain}, [[1,1],[2,2]]] & y^{-1} < x \end{cases} & 0 < y \end{cases}$$

4 Real Roots of Polynomials

Unfortunately, Figure 2, while correct, is an example of problem (i) mentioned in the introduction: in general the boundary cases defining the 'piecewise' construct will not be simple rational numbers, but rather the (real) roots of polynomials. For univariate polynomials, there are a variety of ways of representing such roots.

1. Via an approximation, e.g. "the root of $x^2 - 2$ near 1.5": Maple's `RootOf(x^2-2,x,1.5)`
2. Via a bounding interval, e.g. "the root of $x^2 - 2$ between 1 and 2": Maple's `RootOf(x^2-2,x,1..2)`.
3. Via the signs of derivatives (often described as Thom's Lemma [7]), e.g. "the root of $x^2 - 2$ where $2x$ is positive": this encoding is not supported in Maple.
4. As the i-th complex root ordered in some way, e.g. by increasing argument, as in "the first root of $x^2 - 2$": Maple's `RootOf(x^2-2,x,index=1)`.

Any of these would suffice for polynomials in one variable. However, the polynomial p might be in several variables, e.g. a root in y of a polynomial in x and y, where x lies in a certain region. None of the above methods will then suffice.

*1. The same numeric approximation for y might be close to different branches of p for different x-values.
*2. The same bounding interval approximation for y might be close to different branches of p for different x-values.
*3. The signs of the derivatives might change as we follow a single branch of p.
*4. The order of arguments, or whatever else is used as our order on the complex plane, might change as we follow a single branch of p.

Hence we have been led to suggest extending Maple by a new construct.

5. As the i-th *real* root ordered in some way, e.g. by increasing value, as in "the second real root of $x^2 - 2$": Maple's `RootOf(x^2-2,x,index=real[2])`.

It is a fortunate feature of Maple's design that this works already, in the sense that it is recognized as a root of the polynomial. Nevertheless, work will need to be done to extend, e.g. `evalf`, to understand it properly.

The reader may protest that there is an objection here too.

*5. The number of real roots "before" ours might change as we follow a branch of p, and, for example,the first root might suddenly become the third.

In general this would be a valid objection, but it is a defining characteristic of a *cylindrical* algebraic decomposition that this cannot happen within a given cell, and hence this definition is well-founded *in our application*.

As an example of this construct, we can see the following description of a CAD induced by the polynomial $y^2 - xy - 1$.

$$\begin{cases} [regular\_chain, [[0,0],[-2,-2]]] & y < \texttt{RootOf}(\_Z^2 - x\_Z - 1, index = real[1]) \\ [regular\_chain, [[0,0],[-1,-1]]] & y = \texttt{RootOf}(\_Z^2 - x\_Z - 1, index = real[1]) \\ & And(\texttt{RootOf}(\_Z^2 - x\_Z - 1, index = real[1]) < y, \\ [regular\_chain, [[0,0],[0,0]]] & y < \texttt{RootOf}(\_Z^2 - x\_Z - 1, index = real[2])) \\ [regular\_chain, [[0,0],[1,1]]] & y = \texttt{RootOf}(\_Z^2 - x\_Z - 1, index = real[2]) \\ [regular\_chain, [[0,0],[2,2]]]) & \texttt{RootOf}(\_Z^2 - x\_Z - 1, index = real[2]) < y \end{cases}$$

The reader may think that this is a complicated construction, but a trivial change to `y^2-x*y+1` makes the output too large for this paper, with 17 cases (5 for $x < -2$, 3 for $x = -2$, 1 for $-2 < x < 2$, 3 for $x = 2$ and 5 for $x > 2$) to be considered. The plot is in figure 3.

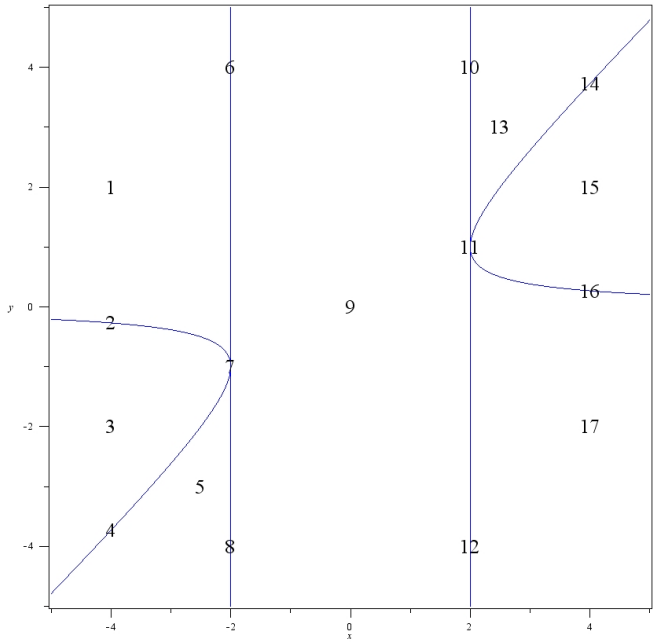
5 Real Triangular Decomposition

The question "what real solutions does a system of polynomial equations have?" does not permit of a simple answer, in general. For an arbitrary input system of equations, inequations and inequalities given by multivariate polynomials, one may wish to describe all solutions of this system which have real coordinates. If there are only finitely many solutions with complex coordinates satisfying the equations and inequations, i.e. we are in *complex* dimension zero, then it is easy, in principle, to detect which ones have real coordinates, and among those, which ones satisfy the inequalities. The output of such process is a list of points with real coordinates, *suitably* encoded.

Our current solution in the case of higher (complex) dimension mimics this: computing a complex triangular decomposition first, and then analyzing the real solutions for each regular system thus produced.

Before considering the general case, let us consider Whitney's umbrella $x^2 - y^2z = 0$ with the variables in the order $x > y > z$. Our code produces the

Fig. 3. Plot for 17-cell CAD



following construct,

$$\begin{cases} [\{x^2 - y^2 z = 0, 0 < z\}] & And\,(y \neq 0, z \neq 0) \\ \texttt{\%RealTriangularize}\left([x^2 - y^2 z, y], [x, y, z], \{\}\right) & And\,(z \neq 0, y = 0) \\ \texttt{\%RealTriangularize}\left([x^2 - y^2 z, z], [x, y, z], \{\}\right) & And\,(y \neq 0, z = 0) \\ \texttt{\%RealTriangularize}\left([x^2 - y^2 z, y, z], [x, y, z], \{\}\right) & \text{otherwise} \end{cases} \tag{3}$$

where all the lower-dimensional cases are left unevaluated. If we force evaluation (using Maple's `value` command), we get

$$\begin{cases} [\{x^2 - y^2 z = 0, 0 < z\}] & And\,(y \neq 0, z \neq 0) \\ \{x = 0, y = 0, z = z\} & And\,(z \neq 0, y = 0) \\ \{x = 0, y = y, z = 0\} & And\,(y \neq 0, z = 0) \\ \{x = 0, y = 0, z = 0\} & \text{otherwise} \end{cases}. \tag{4}$$

Here the first line corresponds to the parametric plane quoted in the introduction, and the second to the "negative z axis". The third component is the fold in the parametric plane, and the fourth the origin, which is the intersection of the components.

Let us now consider the general case. Let us denote by F, H, P the sets of polynomial defining the equations, inequalities and inequalities of the input system. If these polynomials involve n ordered variables $x_1 < \cdots < x_n$, then a solution is any point $\mathsf{r} = (r_1, \ldots, r_n)$ with real coordinates such that $f(\mathsf{r}) = 0$ for all $f \in F$, $h(\mathsf{r}) \neq 0$ for all $h \in H$ and $p(\mathsf{r}) > 0$ for all $p \in P$.

A first idea for solving this system is "simply" to compute a CAD $\mathcal{C}$ of $\mathbf{R}^n$ such that each polynomial of $F \cup H \cup P$ is sign-invariant in each cell C of $\mathcal{C}$. Obviously, this will do much more than we need, since we are not interested, for instance, in the cells where the polynomials of F are strictly negative or strictly positive. This "overkill" is even more obvious in the extreme case where no points satisfy simultaneously all the equations given by F.

A second idea is to proceed in two steps

1. Solve the system of equations given by F over the complex numbers, for instance using triangular decomposition.
2. Filter out these complex solutions (for instance using CAD) to detect those satisfying the inequations and inequalities given by H and P, respectively.

At this point, practical consideration have to be made. Indeed, in many applications, one may first want to obtain the "most likely" or "generic" solutions of a system while keeping the possibility of producing the other solutions later, if necessary. Such motivation has led to weaker versions of the notion of a CAD.

To serve this objective, we propose to combine the piecewise tools of Maple together with its ability of delaying evaluations. Let us see a more interesting example. If we want the *real* solutions of $\{x^2 + y + z - a = 0, x + y + z^2 = 0\}$, we can call

```
sols := RealTriangularize([x^2+y+z-a, x+y+z^2], [x, y, z, a])
```

and get

$$\begin{cases} [\{x + y + z^2 = 0, y^2 + (1 + 2z^2)y + z - a + z^4 = 0 \\ 0 < -z + a + z^2 + 1/4\}] & -4z + 4a + 4z^2 + 1 \neq 0 \\ \texttt{\%RealTriangularize}([x + y + z^2, \\ y^2 + (1 + 2z^2)y + z - a + z^4, -4z + 4a + 4z^2 + 1], [x, y, z, a], \{\}) & \text{otherwise} \end{cases}$$

i.e. some solutions are obtained when $-4z + 4a + 4z^2 + 1 \neq 0$, but determining what happens when $-4z + 4a + 4z^2 + 1 = 0$ requires a (potentially more costly) recursive call, so this has been made inert awaiting explicit evaluation. One could use Maple command `value` to explicitly call `RealTriangularize` and get

$$\begin{cases} \{-1 + 2x = 0, 1 + 4y + 4z - 4a = 0, -4z + 4a + 4z^2 + 1 = 0, a < 0\} & a \neq 0 \\ \texttt{\%RealTriangularize}([-1 + 2x, 1 + 4y + 4z - 4a, \\ -4z + 4a + 4z^2 + 1, a], [x, y, z, a], \{\}) & \text{otherwise} \end{cases}$$

Again, a recursive call is delayed and by `value` one obtains all the solutions represented by a nested piecewise function

$$\begin{cases} \{x + y + z^2 = 0, y^2 + (1 + 2z^2)y + z - a + z^4 = 0 \\ 0 < -z + a + z^2 + 1/4\} & -4z + 4a + 4z^2 + 1 \neq 0 \\ \begin{cases} \{-1 + 2x = 0, 1 + 4y + 4z - 4a = 0, \\ -4z + 4a + 4z^2 + 1 = 0, a < 0\} & a \neq 0 \\ \{a = 0, x = 1/2, y = -3/4, z = 1/2\} & \text{otherwise} \end{cases} & \text{otherwise} \end{cases}$$

6 Further Work

This paper has merely started the discussion of the best way to represent geometrical decompositions such as CTD, CAD, RTD. Several questions remain.

A. More work needs to be done to make `RootOf(p,x,index=real[i])` into a better Maple citizen.

B. The passage from (1) to (2) is currently not automatic. In *many* cases it could be automated, but there are interesting examples where the logic behind comprehensive triangular decompositions is more powerful than that behind `solve`. One limb of (2) is

$$\left\{x = \frac{u^2}{2\,v^2}, y = \frac{-1}{2\,v}\right\} \text{ for } u \neq 0 \text{ and } v^2 + u^3 = 0, \tag{5}$$

where there is an apparent problem for a back-substitution phase if $v = 0$: however this cannot happen because $u \neq 0$ and $v^2 + u^3 = 0$, so $v \neq 0$ is forced.

C. When we computed (1), we used $R := K[x, y, u, v]$ with the argument '2' signifying that this was really $K[x, y; u, v]$ with u and v being parameters. In fact, though, we have also ordered u and v, as required by Triangular Decomposition, though this order is extrinsically meaningless. If we exchange u and v we get the following.

$$\begin{cases} [\{x = 0, y = 0\}, & \\ \{u = 0, x = 0\}] & And\,(v = 0, u = 0) \\ [\{x = 0, y = 0\}, & \\ \left\{1 + 2\,vy = 0, (vy + 1)\,x - u^2y^2 = 0,\right. & \\ \left. v^2 + u^3 = 0, 2\,v \neq 0, vy + 1 \neq 0\right\}] & And\left(v^2 + u^3 = 0, u \neq 0\right) \\ [\{x = 0, y = 0\}, & \\ \left\{(vy + 1)\,x - u^2y^2 = 0, 1 + \left(v^2 + u^3\right)y^2 + 2\,vy = 0,\right. & \\ \left. v^2 + u^3 \neq 0, vy + 1 \neq 0\right\}] & And\left(u \neq 0, v^2 + u^3 \neq 0\right) \end{cases} \tag{6}$$

Since v is the leading parameter, this can be easily specialised with

```
simplify(eval(%,v=0));
```

to the following.

$$\begin{cases} [\{x = 0, y = 0\}, \{0 = 0, x = 0\}] & u = 0 \\ [\{x = 0, y = 0\}, \left\{1 + y^2u^3 = 0, x - u^2y^2 = 0, u^3 \neq 0\right\}] & \text{otherwise} \end{cases} \tag{7}$$

The same problem occurs in Real Triangular Decomposition, where we have asked the user to specify the variable ordering. Much of the time, the user will have a clear idea of what ordering is required, but equally much of the time the user will want the "best" order. Two examples of the same problem with different orders are given above, as (3) (or (4)) and (9)/(10), which are to be contrasted with the description in words given in the introduction.

It is far from clear what "best" means, though the model of simplification in [2] would imply "shortest". There can be radical differences, ranging from polynomial to doubly-exponential, in the size of the output, depending on the order chosen [1]. One promising short-cut for deciding the best order is described in [8]. However, such studies are beyond the remit of this paper, which is to look at the user interface to what is currently being produced. It would be nice, but beyond the scope of this paper, for the system to choose the "best" order of these parameters.

D. As seen in (7), the interaction between `piecewise` and `simplify` could be improved to drop conditions such as the last $u^3 \neq 0$. Equally, we could delete equations which are in the parameters only, reducing (1) to (8).

$$\begin{cases} [\{x=0, y=0\}, \{(vy+1)\,x - u^2y^2 = 0, \\ 1+\left(v^2+u^3\right)y^2 + 2\,vy = 0, vy+1 \neq 0\}] & And\left(u \neq 0, v^2+u^3 \neq 0\right) \\ [\{x=0, y=0\}, \{1+2\,vy = 0, & \\ (vy+1)\,x - u^2y^2 = 0, vy+1 \neq 0\}] & And\left(v^2+u^3 = 0, u \neq 0\right) \\ [\{x=0, y=0\}, \{x=0\}] & u = 0 \end{cases} \tag{8}$$

E. What if the triangular decomposition at the core of RealTriangularize leads to multiple regular chains? Simply returning a sequence of `piecewise` constructs does not really help. Even in simple cases, as occurs in Whitney's umbrella $x^2 - y^2z = 0$ with the variables in the order $z > y > x$, we get the two regular chains (9, which is then displayed as a `piecewise` construct) and (10, which in this case is uniform)

$$\begin{cases} [\{-x^2+y^2z=0\}] & y \neq 0 \\ \texttt{\%RealTriangularize}\left([-x^2+y^2z, y], [z, y, x], \{y^2\}\right) & \text{otherwise} \end{cases}, \tag{9}$$

$$\{x=0, y=0, z=z\} \tag{10}$$

This is just about comprehensible, but clearly this does not scale, and the "pruning" issue referred to above becomes more acute.

It would be possible to approach the decomposition in other ways, e.g. by treating algebraically independent variables as parameters, and doing parametric solving in the first instance, but this is a subject of further research.

F. It could be argued that (4) is redundant, and the equational constraints on the free variables do not need to be repeated. This would give us the following

$$\begin{cases} [\{x^2-y^2z=0, 0<z\}] & And\,(y \neq 0, z \neq 0) \\ \{x=0\} & And\,(z \neq 0, y = 0) \\ \{x=0\} & And\,(y \neq 0, z = 0) \\ \{x=0\} & And\,(y = 0, z = 0) \end{cases} \tag{11}$$

which could then be further simplified to

$$\begin{cases} [\{x^2-y^2z=0, 0<z\}] & And\,(y \neq 0, z \neq 0) \\ \{x=0\} & zy = 0 \end{cases}. \tag{12}$$

7 Conclusion

As can be seen from the CTD, CAD, RTD discussions, by using the *existing* piecewise constructions to describe these geometric concepts, one can design some nice and simple user-interfaces to carry out complex computation of complicated objects, especially for non expert end-users. The use of inert values, another *existing* concept, allows one to hide the *details* of complicated special cases, while *not* hiding their existence. Because they are inert values, the `value` command means the user has access to the details if required.

We are sure that such idea can be applied to some other concepts, like real root classification [10, 11], to build more appropriate user-interfaces from existing components.

References

1. C.W. Brown and J.H. Davenport. The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition. In C.W. Brown, editor, *Proceedings ISSAC 2007*, pages 54–60, 2007.
2. J. Carette. Understanding Expression Simplification. In J. Gutierrez, editor, *Proceedings ISSAC 2004*, pages 72–79, 2004.
3. J. Carette. A canonical form for piecewise defined functions. In C.W. Brown, editor, *Proceedings ISSAC 2007*, pages 77–84, 2007.
4. C. Chen, O. Golubitsky, F. Lemaire, M. Moreno Maza, and W. Pan. *Comprehensive Triangular Decomposition*, volume 4770 of *Lecture Notes in Computer Science*, pages 73–101.
5. C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing Cylindrical Algebraic Decomposition via Triangular Decomposition. To appear in J. May, editor, *Proceedings ISSAC 2009*, ACM Press, New York. `http://arxiv.org/abs/0903.5221`.
6. G.E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, Springer Lecture Notes in Computer Science **33**, pages 134–183, 1975.
7. M. Coste and M.F. Roy. Thom's Lemma, the Coding of Real Algebraic Numbers and the Computation of the Topology of Semi-Algebraic Sets. *J. Symbolic Comp.*, 5:121–129, 1988.
8. A. Dolzmann, A. Seidl, and Th. Sturm. Efficient Projection Orders for CAD. In J. Gutierrez, editor, *Proceedings ISSAC 2004*, pages 111–118, 2004.
9. A.C. Norman. Computing with Formal Power Series. *ACM Transactions on Mathematical Software*, 1:346-356, 1975.
10. L. Yang, X. Hou, and B. Xia. A complete algorithm for automated discovering of a class of inequality-type theorems. *Science in China, Series* **F**, 44(6):33–49, 2001.
11. L. Yang, B. Xia. Real solution classifications of a class of parametric semi-algebraic systems. In: Algorithmic Algebra and Logic — Proceedings of the A3L 2005 (A. Dolzmann, A. Seidl, and T. Sturm, eds.), pp. 281289. Herstellung und Verlag, Norderstedt (2005).