The ConstructibleSetTools and ParametricSystemTools modules of the RegularChains library in Maple

Changbo Chen[1] François Lemaire[2] Liyun Li[1]
Marc Moreno Maza[1] Wei Pan[1] Yuzhen Xie[1]

[1]University of Western Ontario, London ON, N6A 5B7, Canada
[2]Université de Lille 1, 59655 Villeneuve d'Ascq Cedex, France

Abstract

We present two new modules of the RegularChains *library in* Maple*:* ConstructibleSetTools *which is the first distributed package dedicated to the maniputation of (parametric or not) constructible sets and* Parametric-SystemTools *which is the first implementation of comprehensive triangular decomposition. We illustrate the functionalities of these new modules by examples and describe our software design and implementation techniques. Since several existing packages have functionalities related to those of our new modules, we include an overview of the algorithms and software for manipulating constructible sets and solving parametric systems.*

keywords: *triangular decomposition, regular chain, constructible set, parametric polynomial system, comprehensive triangular decomposition, software design*

1. Introduction

Solving systems of equations, algebraic or differential, is a driving subject for symbolic computation. Many practical applications of polynomial system solving require a description of the real solutions of an input system with finitely many complex solutions. Due to the expected shape of the solution set, as studied in [2], this task is typically achieved by means of a Gröbner basis computation followed by the real root isolation of univariate polynomials. Computer algebra systems, such as Maple, develop more and more efficient approaches to this end.

Many other practical and theoretical applications of polynomial system solving require more advanced operations on ideals and varieties, such as *decomposition into components (unmixed, irreducible, ...)*. Primary decomposition of ideals and triangular decomposition of algebraic varieties are concepts which provide the necessary theoretical framework. Algorithms for primary decomposition involve operations on ideals, such as saturation, intersection and quotient computations; their implementation in the computer algebra systems AXIOM, Singular, CoCoA, MAGMA and Maple has led to packages for computing with polynomial ideals, based on Gröbner basis techniques.

The development of triangular decomposition started in the late 80's with the work of W.T. Wu [39], that is, more than 20 years after the introduction of Gröbner bases by B. Buchberger [3]. In the early 90's, the notion of a *regular chain*, introduced independently by M. Kalkbrener in [14] and, by L. Yang and J. Zhang [42], led to important algorithmic progress and stimulated implementation activity.

Up to our knowledge, computer algebra systems provide solvers based on triangular decompositions for 12 years only, mainly in Maple, but also in AXIOM, Singular and MAGMA. Examples of such solvers are the downloadable packages Epsilon by D.M. Wang, WSolve by D.K. Wang, DISCOVERER by B.C. Xia and the RegularChains library [19], which is shipped with Maple since its release 10. Highly efficient solvers based on triangular decomposition are work in progress [21, 22].

This implementation effort is supported by continuous theoretical and algorithmic advances. The notion of *comprehensive triangular decomposition* introduced in [4] has brought to light the fact that *constructible sets* play the role for triangular decompositions that polynomial ideals play for Gröbner bases. This fact was underlying since the early work of W.T. Wu [38]; it became explicit in [4] where the authors provided procedures for computing the difference and the intersection of two constructible sets represented by triangular decompositions. Actually, this work motivated the realization of the software presented in this article.

Comprehensive triangular decomposition (CTD) is one of the tools for parametric system solving, an area which has an increasing number of applications and which is in demand of efficient algorithms and solvers. Of course, the classical techniques based on Gröbner bases and triangular decompositions can process parametric systems. However,

978-0-7695-3243-1/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCSA.2008.61

most practical questions related to these systems require specific theoretical and algorithmic enhancements. To illustrate this fact, let us consider a parametric system $\Sigma(U, X)$ of equations, where U stands for a set of parameters and X for a set of unknowns. For simplicity, we assume that the coefficients of $\Sigma(U, X)$ are in the field $\mathbb{Q}$ of rational numbers and that we are looking for the solutions of $\Sigma(U, X)$ with coordinates in the field $\mathbb{C}$ of complex numbers. A typical problem is to determine the values of U for which $\Sigma(U, X)$ has solutions. This brings the following difficulty: these values of U may not form an algebraic variety, that is, they may not form the solution set of a system of polynomial equations. Consider, for instance, the system $\Sigma(U, X)$ consisting of the single bivariate equation $ux - 1 = 0$, with $U = \{u\}$ and $X = \{x\}$. In this case, the solution set of our problem is given by $u \neq 0$, which is a constructible set, but not an algebraic variety. Therefore, in the context of a strongly typed language, say `AXIOM`, the implementation of a package for parametric systems would naturally imply the implementation of a type "ConstructibleSet". In the case of `Maple`, the implementation of the CTD, as the module `ParametricSystemTools`, has led us to realize a second module, namely `ConstructibleSetTools`. This is why these new modules of the `RegularChains` library are presented jointly in this paper.

Let us consider a more advanced illustrative example from the theory of algebraic curves. Each of the following equations defines an elliptic curve in the complex plane of coordinates (x, y): $g_1(x, y) = 0$ and $g_2(x, y) = 0$, where

$$g_1(x,y) = x^3 + a_1 x - y^2 + 1 \text{ and } g_2(x,y) = x^3 + a_2 x - y^2 + 1.$$

They depend on parameters a_1 and a_2 respectively. In invariant theory, a classical question is whether there exists a linear fractional map from the first curve to the second one:

$$f : (x, y) \mapsto \left(\frac{A\,x + B\,y + C}{G\,x + H\,y + K}, \frac{D\,x + E\,y + F}{G\,x + H\,y + K} \right)$$

See [17] for details. This problem can be turned into a parametric system by writing that the following rational function in (x, y) must be identically zero:

$$g_1(x, y) - (G\,x + H\,y + K)^3 g_2(f(x, y)).$$

This yields a system with 10 equations, 9 unknowns $A, B, C, D, E, F, G, H, K$ and 2 parameters a_1, a_2. Moreover we must have $(G, H, K) \neq (0, 0, 0)$. One can also assume that the origin is mapped to the origin which sets $C = F = 0$. Hence, we have:

$$\left\{ \begin{array}{ll}
1 - K^3 & = 0 \\
-a_2\,AK^2 + a_1 - 3\,GK^2 & = 0 \\
-3\,HK^2 - a_2\,BK^2 & = 0 \\
GD^2 - a_2\,G^2A - A^3 - G^3 + 1 & = 0 \\
-3\,H^2K + E^2K - 1 - 2\,a_2\,BHK & = 0 \\
-3\,G^2K - 2a_2\,GAK + D^2K & = 0 \\
GE^2 - 2a_2\,GBH - a_2\,AH^2 - 3\,AB^2 - 3\,GH^2 + 2\,DEH & = 0 \\
E^2H - H^3 - a_2\,BH^2 - B^3 & = 0 \\
D^2H - 3\,G^2H + 2\,GDE - 2a_2\,GAH - 3\,A^2B - a_2\,G^2B & = 0 \\
-3\,GHK - a_2\,AHK - a_2\,GBK + DEK & = 0
\end{array} \right.$$

The output produced by the command `Comprehensive-Triangularize` of the module `ParametricSystem-Tools` is

```
[regular_chain,regular_chain,regular_chain,regular_chain,
 regular_chain,regular_chain,regular_chain,regular_chain,
 regular_chain,regular_chain,regular_chain],
 [[constructible_set, [1, 2, 3, 10, 11]],
  [constructible_set, [4, 5, 6, 7, 8, 9]],
  [constructible_set, [1, 2, 3]]]
```

It consists of two parts. The first one is a triangular decomposition $dec = [T_1, \ldots, T_{11}]$ of the input system in $\mathbb{Q}[B, E, H, A, D, G, K, F, C, a_1, a_2]$; each regular chain T_j encodes a component of dec. The second one is a list of pairs; in each pair the first item is a constructible set C_i and the second one is a list L_i of indices such that:

1. C_1, C_2, C_3 form a partition of the set of parameter values at which the input system has solutions,
2. each one of the constructible sets C_1, C_2, C_3 is given by a triangular decomposition,
3. for all $i = 1, 2, 3$, the solutions of the input system with parameter values in C_i are given by the regular chains T_j such that $j \in L_i$,
4. for all $i = 1, 2, 3$, for all $j \in L_i$, for all parameter value u in C_i, the regular chain T_j behaves well under specialization at u. (This concept is formally introduced in Definition 6).

We print below the systems defining C_1, C_2, C_3 followed by the regular chains $T_1, \ldots, T_{11}$, as computed by our command.

$$\begin{array}{lcl}
C_1 & : & a_1^3 = a_2^3 = 9 \\
C_2 & : & a_1 = a_2 = 0 \\
C_3 & : & a_1^3 = a_2^3, a_2 \neq 0, a_2^3 \neq 9\,,
\end{array}$$

$$\begin{array}{lcl}
T_1 & = & B, E - 1, H, a_2\,A - a_1, D, G, K - 1, -{a_2}^3 + {a_1}^3 \\
T_2 & = & B, E + 1, H, a_2\,A - a_1, D, G, K - 1, -{a_2}^3 + {a_1}^3 \\
T_3 & = & B, E^2K - 1, H, (a_2\,K + a_2)\,A + a_1, D, G, \\
 & & K^2 + K + 1, -{a_2}^3 + {a_1}^3 \\
T_4 & = & B, E - 1, H, A^2 + A + 1, D, G, K - 1, a_1, a_2 \\
T_5 & = & B, E + 1, H, A^2 + A + 1, D, G, K - 1, a_1, a_2 \\
T_6 & = & B, E - 1, H, A - 1, D, G, K - 1, a_1, a_2 \\
T_7 & = & B, E + 1, H, A - 1, D, G, K - 1, a_1, a_2 \\
T_8 & = & B, E^2K - 1, H, A^2 + A + 1, D, G, K^2 + K + 1, a_1, a_2 \\
T_9 & = & B, E^2K - 1, H, A - 1, D, G, K^2 + K + 1, a_1, a_2 \\
T_{10} & = & a_2\,B + 3\,H, -H a_2\,A + DE, \\
 & & \left(a_2{}^2A^2 - 9\,G^2 + 6\,a_1\,G\right)H^2 + 3\,G^2 - 2\,a_1\,G, \\
 & & -a_1 + a_2\,A + 3\,G, D^2 + 3\,G^2 - 2\,a_1\,G, \\
 & & 9 + 2\,{a_1}^2G, K - 1, {a_1}^3 + 9, {a_2}^3 + 9 \\
T_{11} & = & a_2\,B + 3\,H, -H a_2\,A + DE, \\
 & & \left(3\,D^2K + a_2{}^2A^2K\right)H^2 - D^2, \\
 & & (a_2\,K + a_2)\,A + (3\,K + 3)\,G + a_1, \\
 & & (K + 1)\,D^2 + (3\,K + 3)\,G^2 + 2\,a_1\,G, \\
 & & 2\,{a_1}^2GK - 9\,K - 9, K^2 + K + 1, {a_1}^3 + 9, {a_2}^3 + 9\,.
\end{array}$$

Therefore, the union of the C_i's is the answer to our question: for which parameter values does the input system have solutions. Our software can also compute the union of the C_i's; this will produce a single component, namely

$a_1^3 = a_2^3$; interestingly, these calculations lead to another proof of Theorem 1 in [17].

More generally our software allows the user to perform on constructible sets the usual set theoretical operations: union, intersection, difference, complement, and emptiness-test. More advanced operations such as image (or pre-image) of an algebraic variety by a polynomial map are also available. Actually, the objective of our illustrating example can be stated as to computing the projection of the variety of the input system onto the parameter space.

The paper is structured as follows. Guided by examples, Sections 2 and 3 form an overview of our two new modules of the `RegularChains` library in `Maple`: `ConstructibleSetTools` and `ParametricSetTools`, dedicated respectively to computing with constructible sets and solving parametric systems. Up to our knowledge, `ConstructibleSetTools` is the first distributed package for this purpose. Of course, several existing packages have functionalities related to ours. We attempt to give a survey of those in Section 4. The paper is concluded with a brief discussion on some future research problems.

2. `ConstructibleSetTools` Module

The `ConstructibleSetTools` module is a collection of commands for computing with constructible sets. It consists of routines to build constructible sets, and basic set operations such as `Difference` and `Intersection`, as well as advanced functions including `MakePairwiseDisjoint`, `Projection`, etc.

2.1. Basic Definitions

In what follows, we define the main objects that we manipulate in this package: *regular chain*, *regular system* and *constructible set*. For the details of the related concepts and their properties, please refer to [1], [4] and [26].

Let $\mathbb{K}[X] := \mathbb{K}[X_1, \ldots, X_n]$ be the polynomial ring over the field $\mathbb{K}$ and with ordered variables $X_1 \prec \cdots \prec X_n$. We denote $\overline{\mathbb{K}}$ the algebraic closure of $\mathbb{K}$. For a set of polynomials $F \subset \mathbb{K}[X]$, denote by $V(F)$ the *zero set* (or algebraic variety) of F in $\overline{\mathbb{K}}^n$. For a polynomial $p \in \overline{\mathbb{K}}[X]$, denote by $\text{init}(p)$ the leading coefficient of p regarded as a univariate polynomial in its main variable (the greatest variable). Let $T \subset \mathbb{K}[X]$ be a *triangular set*, that is, a set of non-constant polynomials with pairwise distinct main variables. The saturated ideal $\text{sat}(T)$ of T is defined to be the ideal $\langle T \rangle : {h_T}^\infty$, where h_T is the product of initials of polynomials in T.

Definition 1 (Regular Chain and Quasi-component) *Let T be a triangular set in $\mathbb{K}[X]$. If T is empty, then it is a regular chain. Otherwise, let p be the polynomial of T with the greatest main variable and let C be the set of other polynomials in T. We say that T is a* regular chain*, if C is a* regular chain *and* $\text{init}(p)$*, the initial of p, is regular modulo* $\text{sat}(C)$*. In addition, the* quasi-component $W(T)$ *of T is defined to be $V(T) \setminus V(h_T)$, where h_T is the product of the initials of the polynomials in T.*

Definition 2 (Regular System) *A pair $[T, h]$ is a* regular system *if T is a regular chain, and $h \in \mathbb{K}[X]$ is regular with respect to* $\text{sat}(T)$*. The zero set $Z(T, h)$ given by $[T, h]$ is $W(T) \setminus V(h)$.*

Definition 3 (Constructible Set) *A* constructible set *of $\overline{\mathbb{K}}^n$ is a finite union $(A_1 \setminus B_1) \cup \cdots \cup (A_e \setminus B_e)$ where $A_1, \ldots, A_e, B_1, \ldots, B_e$ are algebraic varieties in $\overline{\mathbb{K}}^n$.*

Proposition 1 ([4]) *The zero set of any regular system is unmixed and nonempty. Every constructible set can be written as a finite union of the zero sets of regular systems.*

2.2. Software Design

Our software design mimics the organization of categories and domains in the computer algebra system `AXIOM` [13]. Both regular systems and constructible sets are treated as classes of objects, *regular_system* and *constructible_set* . This follows the implementation strategy of the `RegularChains` library [20], where a regular chain is a class type *regular_chain*. This implementation technique enhances the extensibility and reusability of our code.

The `RegularChains` user-interface has been organized into two-level modules. The `ConstructibleSetTools` module lies in the second-level interface. It relies on the top-level module `RegularChains` and the `ChainTools` submodule as back-end engines.

We also provide flexibility in viewing the data. Since symbolic computation generally involves large expressions, our functions are devised to display their computing results as the types of the objects which they represent, i.e. *regular_chain*, *regular_system* or *constructible_set* . The user can then chose to view more details by our displaying tools such as `Equations` and `Info`.

A special design issue is on *irredundancy and lazy evaluation*. Let $Z(A)$ denote the zero set of A, where A can be either a regular system or a constructible set. Assume that $[A_1, \ldots, A_s]$ is a list of regular systems composing a constructible set. For $i, j \in \{1, \cdots, s\}$ and $i \neq j$, $Z(A_i) \cap Z(A_j)$ may not be empty. In the functions to build a constructible set, a decision has to be made for whether or not to have $Z(A_i) \cap Z(A_j) = \emptyset$ hold. The same question arises for all the basic operations on constructible sets. Ideally one would like to have irredundant objects in the computations all along. However, the operation to remove

these redundant objects is not cheap. Frequent calls to this operation can even be a bottleneck to the overall process.

In this package, we apply a *lazy evaluation* strategy. Removing redundant objects is not considered at all either when constructing a constructible set or in the basic operations. Instead, we provide an extra function named `MakePairwiseDisjoint`. It takes a constructible set as input and makes the zero sets of its defining regular systems pairwise disjoint. In case irredundant results are demanded, the function `MakePairwiseDisjoint` can be used to clean the results by paying extra cost.

2.3. Creating Constructible Sets

In this subsection, we illustrate by examples how to create constructible sets in different ways. Always, we read in necessary modules and define a polynomial ring to set up the space we are working on.

```
> with(RegularChains): with(ChainTools):
> with(ConstructibleSetTools):
> R := PolynomialRing([x,y,z],0):
```

By default, R is defined to be $\mathbb{Q}[x \succ y \succ z]$. The second argument 0 indicates that the ring characteristic is zero.

Example 1 (The empty constructible set)

```
> cs1 := EmptyConstructibleSet(R);
```

$$cs1 := constructible\_set$$

```
> IsEmpty(cs1,R);
```

$$true$$

The set cs_1 created above is the unique empty set of $\overline{\mathbb{Q}}^3$. The emptiness of a constructible set can be checked via the command `IsEmpty`.

Example 2 (Quasi-component of a regular chain)

The quasi-component of a regular chain is also a constructible set, one can make such a conversion by the function `QuasiComponent`. Given a system of polynomials F, we first use the command `Triangularize` to decompose F into a list of regular chains.

```
> F := [x*y*z-x*y,y^2-y*z];
```

$$F := [xyz - xy, y^2 - yz]$$

```
> dec := Triangularize(F,R);
```

$$dec := [regular\_chain, regular\_chain, regular\_chain]$$

As we mentioned before, only the type of the output will be displayed. The procedure `Info` can be used to retrieve the internal defining polynomials of a regular chain, a regular system or a constructible set. For example, we can apply it to the second regular chain in dec.

```
> rc := dec[2]; Info(rc,R);
```

$$rc := regular\_chain$$
$$[x, y - z]$$

Now we are ready to build the quasi-component of rc.

```
> cs2 := QuasiComponent(rc,R);Info(cs2,R);
```

$$cs2 := constructible\_set$$
$$[[x, y - z], [1]]$$

Here $cs2$ is defined by one regular system in which the defining regular chain is rc and there is no other inequation (we put '[1]' there for this purpose), as we expected.

Example 3 (Creating a regular system)

Following the definition of a regular system, a function `RegularSystem` is provided to construct a regular system from a regular chain with a list of polynomials as inequations. Here each polynomial is assumed to be regular w.r.t. the regular chain. To check this, one can use `IsRegular` which is a function in the `ChainTools` module.

We use those regular chains created in the last example. Suppose that we are interested in those points each of which is in the quasi-component of rc but its y-coordinate is not zero. First we check if y is regular w.r.t. rc.

```
> IsRegular(y,rc,R);
```

$$true$$

Thus we can build a regular system from rc and y.

```
> rs := RegularSystem(rc,[y],R);
```

$$rs := regular\_system$$

```
> Info(rs,R);
```

$$[[x, y - z], [y]]$$

The regular system rs created above will encode the exact points we are looking for.

Example 4 (Creating a constructible set)

A constructible set can be represented by a finite list of regular systems. The command `ConstructibleSet` can be used to create this main object.

We reuse the regular chains in Example 2. First, we construct a list of regular systems from a list of regular chains. For each point in these quasi-components, we impose that its y-coordinate does not equal to 2. That is to say, we add $y-2$ into the inequation part to create new regular systems.

```
> lrs := map(RegularSystem,dec,[y-2],R);
```

$$lrs := [regular\_system, regular\_system, regular\_system]$$

```
> map(Info,lrs,R);
```

$$[[[y], [1]], [[x, y - z], [y - 2]], [[y - 1, z - 1], [1]]]$$

Note that there exist some simplifications during building regular systems. For example, in the first component, y is zero and while trying to add $y-2$ into the inequation part, the constructor will ignore this inequation once it detect such an obvious simplification. The function `ConstructibleSet` is used to create a constructible set from a list of regular systems.

```
> cs3 := ConstructibleSet(lrs, R);
```

$$cs3 := constructible\_set$$

```
> Info(cs3,R);
```
$$[[y],[1]],\ [[x, y-z],[y-2]],\ [[y-1, z-1],[1]]$$

Example 5 (General Construct)

We provide a very synthetic way to create a constructible set via the command `GeneralConstruct` which accepts a list of polynomials as equations, a regular chain and a list of polynomials as inequations. It constructs a constructible set which encodes all the points satisfying these constraints. This function can be regarded as a generalized `Triangularize` command.

Let F be the polynomial system defined in Example 2. As follows, one can create a constructible set, each point in which cancels all polynomials in F and its y-coordinate is nonzero.

```
> cs4 := GeneralConstruct(F,[y],R);
```
$$cs4 := constructible\_set$$
```
> Info(cs4,R);
```
$$[[x, y-z],[z]],\ [[y-1, z-1],[1]]$$

Note that this function can accept different kinds of input forms. Here the regular chain by default is empty.

2.4. Basic Operations on Constructible Sets

In this subsection, we continue introducing functions on some set theoretical operations with constructible sets. We define a polynomial ring R for the following examples.

```
> R := PolynomialRing([x,y,u,v]):
```

Through the following sequence of commands, we build two constructible sets $cs1$ and $cs2$.

```
> G := [x^2+y^2-1, u*x-v*y];
```
$$G := [x^2+y^2-1, ux-vy]$$
```
> cs1 := GeneralConstruct(G,[x],R);
```
$$cs1 := constructible\_set$$
```
> Info(cs1,R);
```
$$[[ux-vy, (u^2+v^2)y^2-u^2],[y,v]]$$
$$[[x-1,y,u],[1]],\ [[x+1,y,u],[1]],\ [[x^2+y^2-1,u,v],[x]]$$
```
> cs2 := GeneralConstruct(G,[y],R);
```
$$cs2 := constructible\_set$$
```
> Info(cs2,R);
```
$$[[ux-vy, (u^2+v^2)y^2-u^2],[y]],\ [[x^2+y^2-1,u,v],[y]]$$

Example 6 (Difference) *The difference of two constructible sets is again a constructible set.*

```
> cs3 := Difference(cs1,cs2,R);
```
$$cs3 := constructible\_set$$
```
> Info(cs3,R);
```
$$[[x-1,y,u],[v]],\ [[x-1,y,u,v],[1]],$$
$$[[x+1,y,u],[v]],\ [[x+1,y,u,v],[1]]$$

Based on `Difference`, the module provides a function called `IsContained` which can be used to detect if one constructible set is contained in another one.

```
> IsContained(cs3,cs1,R);
```
$$true$$

Example 7 (Union) *The command* `Union` *will form the union of two constructible sets, simply by putting all defining regular system together.*

```
> cs6 := Union(cs1,cs2,R); Info(cs6,R);
```
$$cs6 := constructible\_set$$
$$[[ux-vy, (u^2+v^2)y^2-u^2],[y,v]],$$
$$[[x-1,y,u],[1]],\ [[x+1,y,u],[1]],\ [[x^2+y^2-1,u,v],[x]],$$
$$[[ux-vy, (u^2+v^2)y^2-u^2],[y]],\ [[x^2+y^2-1,u,v],[y]]$$

Due to our lazy evaluation strategy, the output of `Union` may contain redundancy. For example, the regular system

$$[[ux-vy, (u^2+v^2)y^2-u^2],[y,v]]$$

is contained in the regular system

$$[[ux-vy, (u^2+v^2)y^2-u^2],[y]].$$

If an irreduntant result is demanded, one can call `MakePairwiseDisjoint` to clean $cs6$.

```
> cs7 := MakePairwiseDisjoint(cs6,R);
```
$$cs7 := constructible\_set$$
```
> Info(cs7,R);
```
$$[[x^2+y^2-1,u,v],[y]],\ [[x-1,y,u],[1]],$$
$$[[x+1,y,u],[1]],\ [[ux-vy, (u^2+v^2)y^2-u^2],[y]]$$

The constructible set $cs7$ encodes the same set of points as $cs6$, but the zero sets of its defining regular systems are pairwise disjoint.

Example 8 (Refining Partition)

There is another level of redundancy: among a list of constructible sets in_lcs, there exists intersection between two constructible sets in this list. In this case, there is a set-theoretical co-prime factorization problem: constructing another finite list of *pairwise disjoint* constructible sets out_lcs such that every constructible set in in_lcs can be *uniquely* written as a union of several constructible sets in out_lcs. This task is achieved by the command `RefiningPartition`.

```
> rp := RefiningPartition([cs1,cs2],R);
```
$$rp := \begin{bmatrix} constructible\_set & [2] \\ constructible\_set & [2,1] \\ constructible\_set & [1] \end{bmatrix}$$

The above output is represented by a matrix in which the first column are constructible sets and the second column

are indices showing where the constructible sets come from. In the above matrix, the constructible set in the first row with index "2" is $cs2 \backslash cs1$, the second one with index "1, 2" is $cs1 \cap cs2$ and the third one with index "1" is $cs1 \backslash cs2$. In fact, the three constructible sets in rp is an "intersection-free" bases of $cs1$ and $cs2$.

2.5. Two Advanced Operations

In this subsection, we show two more advanced operations related to constructible sets.

The `Projection` function is used to project a variety onto its coordinate space, where the result is not necessary a variety but a constructible set. This function is based on a key operation for computing pre-comprehensive triangular decompositions in the `ParametricSystemTools` module which will be introduced in next section.

Example 9 (Projection of a variety)

```
> R := PolynomialRing([x,a,b,c]);
  f := a*x^2+b*x+c;
  cs := Projection([f],3,R); Info(cs,R);
```

$$R := \textit{polynomial\_ring}$$

$$f := ax^2 + bx + c$$

$$cs := \textit{constructible\_set}$$

$$[[\,], [a]],\ [[a], [b]],\ [[a, b, c], [1]]$$

In this example, we project the variety defined by f onto the parameter space defined by the last 3 variables a, b, c. The projection image cs is a constructible set which consists of three components:

$$(1)\ a \neq 0; (2)\ a = 0, b \neq 0; (3)\ a = b = c = 0.$$

These results describe for what values of a, b, c, the equation $f = 0$ has a solution over $\mathbb{C}$.

We next compute the complement of cs, obtaining a constructible set $cs1$. By $cs1$ we conclude that for the case where $a = b = 0$ and $c \neq 0$, f has no solutions over $\mathbb{C}$.

```
> cs1 := Complement(cs, R); Info(cs1,R);
```

$$cs := \textit{constructible\_set}$$

$$[[a, b], [c]]$$

Remark 1 *The output of* `Projection(F, d, R)` *addresses the question "for what values of the last* d *variables regarded as parameters over the polynomial ring* R*, does the parametric system* F *have solutions?"*

A frequent application in *Computer Graphics* is the computation of images and preimages of algebraic varieties under polynomial maps. The images and preimages do not have to be algebraic varieties but constructible sets. We provide functions `PolynomialMapImage` and `PolynomialMapPreimage` in this module to compute these objects. The following example illustrates how to use the function `PolynomialMapImage`.

Example 10 (Polynomial map image)
To begin with, we define the source space S *and the target space* T *where the image lives in. The polynomial map* PM *is defined as a list of polynomials in* S*.*

```
> S := PolynomialRing([t]);
>   T := PolynomialRing([x,y,z]);
>   PM := [t,t^2,t^3];
```

$$S := \textit{polynomial\_ring}$$

$$T := \textit{polynomial\_ring}$$

$$PM := [t, t^2, t^3]$$

```
> cs1 := PolynomialMapImage([],PM,S,T);
>   Info(cs1,T);
```

$$cs1 := \textit{constructible\_set}$$

$$[[-z + xy, y^3 - z^2], [y]],\ [[x, y, z], [1]]$$

The call to `PolynomialMapImage`$([\,], PM, S, T)$ computes the image $cs1$ under the polynomial map PM in the three-dimensional space. As a constructible set, $cs1$ is composed of two regular systems. The two regular systems together represent the variety defined by

$$-z + xy = 0 \text{ and } y^3 - z^2 = 0,$$

which describes the well-known *twisted cubic*.

3. `ParametricSystemTools` Module

The `ParametricSystemTools` module is a collection of commands for solving polynomial systems depending on parameters. It is a direct application of the `ConstructibleSetTools` module presented in the previous section.

The main commands included in this module are

- `DefiningSet`,
- `ComprehensiveTriangularize`,
- `PreComprehensiveTriangularize`,
- `DiscriminantSet`,
- `Specialize`.

These functions can be used to understand the properties of the solution set of a polynomial system F which depends on parameters. For instance, you can answer questions like: for which values of the parameters does F have solutions? finitely many solutions, or N solutions for a given $N > 0$?

The functions of this module are based on the concept of *comprehensive triangular decomposition (CTD)* for a parametric polynomial system with coefficients in a field. This notion plays the role for triangular decompositions that comprehensive Gröbner basis [36, 25] does for Gröbner bases. A special feature of CTD is that *bad specializations* can be avoided. Indeed, CTDs have a motivation other than counting solutions depending on parameters. This feature can be stated as follows: above each cell in the parameter space, each regular chain in the associated triangular decomposition must *behave well* under specializations. This concept will be illustrated in detail later in this section.

3.1. Basic Definitions

Let F be a finite set of polynomials with coefficients in $\mathbb{K}$, parameters $U = U_1, \ldots, U_d$, and unknowns $X = X_1, \ldots, X_m$, that is $F \subset \mathbb{K}[U_1 \prec \cdots \prec U_d \prec X_1 \prec \ldots \prec X_m]$. Let $\overline{\mathbb{K}}$ be the algebraic closure of $\mathbb{K}$, and let π_U be the projection from $\overline{\mathbb{K}}^{d+m}$ on the parameter space $\overline{\mathbb{K}}^d$. For each $u \in \overline{\mathbb{K}}^d$ we define $V(F(u)) \subseteq \overline{\mathbb{K}}^m$ the zero set defined by F after specializing U at u.

Definition 4 (Specialize Well and Defining Set) *Given a positive integer d, a regular chain T can be splitted into two parts. Denote by T_0 the set of the polynomials in T involving only the last d variables, and denote by T_1 the other polynomials of T. Let W be the quasi-component of T_0.*

The regular chain T specializes well *at a point u of W if $T_1(u)$ is a regular chain after specialization and no initial of polynomials in T_1 vanishes during the specialization. The* defining set *of T with respect to the last d variables consists of the points in W at which T specializes well.*

Remark 2 *For a point u in W, after specializing T_1 at u, two situations arise: either T_1 is not a regular chain anymore; or T_1 is still a regular chain. There is a subtle point: after specializing T_1 at u, it might happen that it is still a regular chain, but its shape changes. In other words, the degree of the geometric object given by T_1 could change. The term* specialize well*, defined above, takes these cases into account.*

In what follows, we illustrate the functionalities of the `ParametricSystemTools` module by examples.

The function `DefiningSet` computes the defining set of a regular chain T with respect to the last d number of variables regarded as parameters. This is a constructible set consisting of points in the quasi-component of T_0 at which T_1 specializes well.

In the following example, F is a set of polynomials in $\mathbb{Q}[x, y, u, v]$. For different values of u and v, the solution set has different nature. For example, $u = 0$ can be seen as a degenerate case, in which $x = 0$ and y can be any value. To understand better on $V(F)$, we first decompose F into a list c of regular chains in the sense of Lazard [26], where all solutions of F will be given by the quasi-components of these regular chains.

Next, we call `DefiningSet`$(c[1], 2, R)$ to find out the defining set $ds1$ of the first regular chain $c[1]$ with respect to the last 2 parameters u and v. The content of $ds1$ shows that $c[1]$ is well-specialized for all values of u and v. However, for the last regular chain $c[4]$, its defining set is given by $u^3 + v^2 = 0$ and $v \neq 0$, and the inequation is to assure that the regular chain $c[4]$ specializes well.

Example 11 (Defining set of a regular chain)

```
> R := PolynomialRing([x,y,u,v]):
F := [v*x*y+u*x^2+x,u*y^2+x^2]:
c := Triangularize(F,R,output=lazard);
map(Info,c,R);
```

$$c := [regular\_chain, regular\_chain, regular\_chain, regular\_chain]$$

$$[[x, y], [x, u],$$

$$[(vy + 1)\, x - u^2y^2, 1 + \left(u^3 + v^2\right) y^2 + 2\, vy],$$

$$[(vy + 1)\, x - u^2y^2, 1 + 2\, vy, u^3 + v^2]]$$

```
> ds1:=DefiningSet(c[1],2,R);Info(ds1,R);
```

$$ds1 := constructible\_set$$

$$[[\,], [1]]$$

```
> ds4:=DefiningSet(c[4],2,R);Info(ds4,R);
```

$$ds4 := constructible\_set$$

$$[[u^3 + v^2], [v]]$$

It can happen that the defining set of a regular chain T is strictly contained in the projection of $W(T)$ on $\overline{\mathbb{K}}^d$, that is, T may not specialize well at some points of $\overline{\mathbb{K}}^d$. This difficulty is overcome by the notion of *pre-comprehensive triangular decomposition* (PCTD). More formally, a triangular decomposition $pctd$ of a parametric system F is pre-comprehensive if for all parameter value u: the solution set of $F(u)$ is the union of the $W(T(u))$ for all T in $pctd$ such that u belongs to the defining set of T.

Definition 5 *A* pre-comprehensive triangular decomposition *of $V(F)$ is a family of regular chains $\mathcal{T}$ satisfying the following property: for each $u \in \overline{\mathbb{K}}^d$, let $\mathcal{T}_u$ be the subfamily of all regular chains in $\mathcal{T}$ that specialize well at u; then*

$$V(F(u)) = \bigcup_{T \in \mathcal{T}_u} \mathbf{W}(T(u)).$$

In the example below, F is a list of polynomials in $\mathbb{Q}[x, y, s]$. Its triangular decomposition consists of two regular chains. If $s = 0$, then the first regular chain, says $[(y+1)x - s, y^2 - s + y]$, specializes to $[(y+1)x, y^2 + y]$, which is not a regular chain since the initial $y+1$ of $(y+1)x$ is a zerodivisor w.r.t $y^2 + y$. This difficulty is resolved by the call to `PreComprehensiveTriangularize(F,1,R)`, which computes a PCTD of F regarding the last variable s as a parameter. There are three regular chains in the result $pctd$. For all value s, the solution set of $F(s)$ is the union of the $W(T(s))$ for all T in $pctd$ such that s belongs to the defining set of T.

Example 12 (PCTD)

```
> R := PolynomialRing([x,y,s]):
F := [s-(y+1)*x,s-(x+1)*y]:
dec := Triangularize(F,R,output=lazard);
map(Info,dec,R);
```

$$dec := [regular\_chain, regular\_chain]$$

$$[[(y + 1)\, x - s, y^2 - s + y], [x + 1, y + 1, s]]$$

```
> pctd :=
PreComprehensiveTriangularize(F,1,R);
map(Info,pctd,R);
```

$$pctd := [regular\_chain, regular\_chain, regular\_chain]$$

$$[[(y+1)\,x - s, y^2 - s + y], [x+1, y+1, s], [x, y, s]]$$

Remark 3 *How is* `Projection` *function implemented?*

The `Projection(F, d, R)` *function computes the projection image of $V(F)$ on the parameter space defined by the last d variables. The essence of this command is to compute a pre-comprehensive triangular decomposition $pctd$ of F regarding the last d variables as parameters, and then compute the union of the defining sets of all regular chains in $pctd$.*

Comparing with the notion of pre-comprehensive triangular decomposition, that of *comprehensive triangular decomposition (CTD)* provides additional information on the geometry of the input parametric system F after specialization. In broad terms, a CTD is a partition of the parameter space such that above each part (or cell) the geometry of $V(F)$ is constant. More formally, a CTD comprises a finite partition of the parameter space into cells (each of them given by a constructible set) and for each cell C, a family of regular chains all specialize well at any point of C and such that the zeroes of F above C are exactly described by the quasi-components of those regular chains.

Definition 6 (Comprehensive Triangular Decomposition) *A CTD of $V(F)$ is given by :*

1. *a finite partition $\mathcal{C}$ of $\pi_U(V(F))$,*
2. *for each $C \in \mathcal{C}$ a set of regular chains $\mathcal{T}_C$ of $\mathbb{K}[U, X]$ such that for $u \in C$ each of the regular chains $T \in \mathcal{T}_C$ specializes well at u and we have for all $u \in C$*

$$V(F(u)) = \bigcup_{T \in \mathcal{T}_C} W(T(u)).$$

The example below illustrates how the command `ComprehensiveTriangularize` is used for computing a CTD of a parametric system

$$F = [x^2 + y^2 - 1, ux - vy]$$

in $\mathbb{Q}[x \succ y \succ u \succ v]$ with u and v as parameters.

The output consists of two parts, $pctd$ and $cells$. $pctd$ is a pre-comprehensive triangular decomposition of F. $cells$ is a list of constructible sets with indices. All the constructible sets together form a partition of the parameter space defined by u and v. The list of indices attached to a constructible set indicates that this constructible set is contained in the defining set of the regular chains located in $pctd$ by these indices. In other words, above a constructible set C in $cells$, the zeroes of F above C are represented by the union of the quasi-components of the regular chains in the list of $pctd$ indexed by the indices related to C. For instance, the union of the quasi-components of regular chains $pctd[2]$ and $pctd[3]$, which are $[x-1, y, u]$ and $[x+1, y, u]$, describes exactly the zeroes of F for the case where $u = 0$ but $v \neq 0$, given by the second constructible set in $cells$.

Example 13 (CTD)

```
> R := PolynomialRing([x,y,u,v]):
F := [x^2+y^2-1, u*x-v*y]:
pctd,cells :=
ComprehensiveTriangularize(F,2,R);
```

$$pctd,\ cells := [regular\_chain, regular\_chain,$$

$$regular\_chain, regular\_chain],\ [[constructible\_set, [1]],$$

$$[constructible\_set, [2, 3]], [constructible\_set, [2, 3, 4]]]$$

```
> map(Info,pctd,R);
```

$$[[ux - vy, (u^2 + v^2)\,y^2 - u^2], [x-1, y, u],$$

$$[x+1, y, u], [x^2 + y^2 - 1, u, v]]$$

```
> for e in cells do Info(e[1], R);
  end do;
```

$$[[], [u, u^2 + v^2, v]],\ [[v], [u]]$$

$$[[u], [v]]$$

$$[[u, v], [1]]$$

The `ParametricSystemTools` module provides a function `DiscriminantSet` for determining the discriminant set of a parametric polynomial system.

Definition 7 (Discriminant Set) *The discriminant set of F is defined as the set of all points $u \in \overline{\mathbb{K}}^d$ for which $V(F(u))$ is empty or infinite.*

Example 14

```
> R := PolynomialRing([x,y,u,v]):
  F := [x^2+y^2-1,u*x-v*y]:
  cs := DiscriminantSet(F,2,R);
  Info(cs,R);
```

$$cs := constructible\_set$$

$$[[u, v], [1]],\ [[u^2 + v^2], [v]]$$

4. Related Work

In this section, we describe some basic routines implemented in our two new modules. Meanwhile, we summarize related algorithms, implementations and packages.

4.1. Related Work for Manipulating Constructible Sets

Recall that a constructible set is a finite union of subsets $W = A \setminus B$ with A and B being varieties. Different representations for W gives quite different methods to realize basic operations like difference, intersection, and projection.

Gröbner basis approach The above two varieties A and B can be given by two reduced Gröbner bases, as in O'Halloran and Schilmoeller [27], Kemper [16], Schauenburg [28], Montes [23], etc. In [23], as part of computing canonical comprehensive Gröbner systems, the authors proposed an algorithm to describe the segments (constructible sets) in a canonical way, where a constructible set is represented by a P-tree (each node, except the root, is a prime ideal) with some additional properties.

The problem of computing the difference or intersection of two constructible sets boils down to manipulating polynomial ideals like computing the intersection of two ideals. In [29], the author proposed algorithms for testing inclusion and equality relations between two constructible sets. The main technique is based on radical membership tests with Gröbner basis. Similar technique was also used in [6] to make the constructible subsets of parameter space consistent. The projection of a variety may be seen as a refinement of classical elimination and extension theorem [9]. In [28], the authors describe an algorithm to compute the projection image of a variety. This approach is geometrical and there exists certain analogy with the one based on the notion of well-specialization in this paper.

Triangular set approach By the technique of triangular decompositions [38], the set W can be decomposed into the union of zero sets of triangular systems [34, 7]. Each triangular system is a pair $[T, h]$, where T is a triangular set and h is a polynomial. The points encoded by the pair $[T, h]$ are $V(T) \setminus V(h)$. A very first problem is that this set may be empty, and there are several constraints added to a triangular system.

In [8, 11], the triangular set is normalized, that is the initials only involve its parameters. In [33, 34], the concept (Wang) regular system was proposed in which both the triangular set and the inequations are normalized, and all initials of the defining polynomials will not vanish. The complexity result in [10] shows that while decomposing a polynomial system into a sequence of normalized triangular sets the output size is worse than that of into a sequence of regular chains. Therefore, in [4] the authors defined the notation of a regular system as in Definition 2. Since the zero set of a regular system is always nonempty (in fact unmixed), by Proposition 1 a constructible set is empty if and only if it is given by an empty list of regular systems. This is a key reason that we choose the regular system representation for a constructible set.

Although the difference of two constructible sets can be computed naively, that is, reduce the problem to compute the union or intersection of two varieties. In [5], an efficient algorithm was developed for computing the difference of the zero sets of two regular systems. The basic idea there is to make better use of the triangular structure of the input. A similar idea applies to compute the intersection.

To compute the projection of a constructible set, one may first decompose it into the union of zero sets of triangular systems. Then it is enough to compute the projection of the zero set of each triangular system. To do this, one can apply pseudo-division to eliminate variables of a triangular system one by one. Such an approach was first proposed by Wu [40] and then further developed by other authors [7, 34, 8]. There is an interesting result stated in [34, 35], which pointed out that the projection of the zero set of a (Wang) regular system is simply the common zeros of some polynomials in the regular system. This property is called the strong projection property of a (Wang) regular system in [35]. Despite of this good property, the cost to maintain the strong projection property is high.

4.2 Related Work for Parametric Polynomial System Solving

As a special kind of polynomial system, parametric systems of linear equations were studied carefully in [30]. There the author gave an efficient algorithm for identifying all choices of parametric values for which the system is solvable, and for each choice, solve the linear system.

In [36] and [8], general parametric polynomial systems were considered via comprehensive Gröbner bases and triangular set approaches respectively. In [15], by virtue of constrained polynomials, the author proposed two similar approaches for solving parametric polynomial equations, which are parametric Gröbner basis and parametric characteristic set. The concept of parametric Gröbner basis was further developed by [6] into Partitioned-Parametric Gröbner bases (PPGB). In [12], the Hilbert function was used as a tool for checking the specialization of a parametric Gröbner basis. Recently, a new technique based on Gröbner basis with block term ordering was proposed in [18], where the author investigated for which parameters a polynomial system is well-behaved by computing the minimal discriminant variety.

Following the direction of comprehensive Gröbner bases, different algorithms for computing comprehensive Gröbner bases (CGB) and comprehensive Gröbner systems (CGS) have been proposed, such as CGB [36], BUILDTREE (previously called DISPGB) [25, 24, 23], CCGB [37], ACGB [31], SACGB [32], etc. Comparing with comprehensive triangular decomposition, CGB or CGS maintains more algebraic information such as multiplicity while CTD provides more geometrical information such as degree and unmixed dimension.

Following the direction of triangular decompositions, there are also lots of work [41, 8, 11, 35] devoted to parametric polynomial systems solving. In [41] the authors studied parametric semi-algebraic sets whereas for other papers the parametric polynomial system was the object of

study. The concept cover proposed in [8] is very similar to pre-comprehensive triangular decomposition (PCTD) in [4]. However, the latter is based on the specializing well property of a general regular chain while the former relies on that of normalized regular chain.

There are a few other packages or programs available related to parametric system solving and manipulating constructible sets, such as `DPGB`, `DV`, `QuasiAlgebraicSet`, `RootFinding[Parametric]`, `SACGB`, `DISCOVERER`, `Epsilon`, `WSolve`, etc. The first five packages are based on Gröbner basis approach while the rest three are implemented via triangular decompositions. Except `QuasiAlgebraicSet`, which is implemented in `AXIOM`, all the packages are implemented in `Maple`.

Although there are many related implementations, to our knowledge, there is no package which dedicates to manipulating constructible sets in a systematic way before the development of our `ConstructibleSetTools` module. On the front of solving parametric polynomial systems, our `ParametricSystemTools` module provides different features compared with tools based on Gröbner basis approach. Among the packages based on triangular decompositions, `DISCOVERER` focuses on real solving; `WSolve` does not have dedicated functions for parametric system solving; `Epsilon` can be applied to parametric polynomial system solving, but it usually computes more than needed since the parameters are not prescribed. Our `ParametricSystemTools` module explicitly distinguishes variables and parameters and is a tool particularly aiming at parametric polynomial system solving.

4.3. A Small Comparison with SACGB

We report here a brief comparison of our function `ComprehensiveTriangularize` with Akira Suzuki and Yosuke Sato's `Maple` implementation SACGB on comprehensive Gröbner basis [32]. Another comparison between `ComprehensiveTriangularize`, RegSer and DISPGB can be found in [4].

Table 1 illustrates the timing and the length of the output regarded as a string on the 6 examples from [32]. The tests are performed in `Maple` 9.5 on an Intel Pentium 4 machine (2.60GHz CPU, 1.0GB memory).

	SACGB			CTD		
Sys	Time(s)	# Segs	Length	Time(s)	# Cells	Length
1	38.1	13	120788	5.8	4	591
2	Error	–	–	$>$ 1 hour	–	–
3	$>$ 1 hour	–	–	$>$ 1 hour	–	–
4	904.7	27	23398	2.0	9	958
5	$>$ 1 hour	–	–	1.8	7	961
6	$>$ 1 hour	–	–	1.3	5	864

Table 1 Comparison with SACGB on 6 examples

Here we list the defining polynomials of cells (segments) on the parameter space for the first example in Table 1. The output by SACGB consists of the following 13 segments:

$$
\begin{aligned}
&C_1 : b = 0, a \neq 0\\
&C_2 : b = 0, a^4 = 0, a \neq 0\\
&C_3 : b^3 = 0, a = 0, b \neq 0\\
&C_4 : 729a^4 + 64b^3 = 0, ab \neq 0\\
&C_5 : b = 0, a^8 = 0, a \neq 0\\
&C_6 : b^6 = 0, a = 0, b \neq 0\\
&C_7 : 16767a^4 + 5632b^3 = 0, b^6 = 0, ab \neq 0\\
&C_8 : a = 0, b \neq 0\\
&C_9 : b = 0, a = 0\\
&C_{10} : ab(16767a^4 + 5632b^3)(-4096b^3 + 729a^4)(729a^4 + 64b^3)^2 \neq 0\\
&C_{11} : -4096b^3 + 729a^4 = 0, ab \neq 0\\
&C_{12} : ab(16767a^4 + 5632b^3)(-4096b^3 + 729a^4)(729a^4 + 64b^3) \neq 0,\\
&\qquad -2939328b^3a^4 - 262144b^6 + 531441a^8 = 0\\
&C_{13} : 16767a^4 + 5632b^3 = 0, ab \neq 0.
\end{aligned}
$$

The partition `ComprehensiveTriangularize` contains the 4 nonempty cells,

$$
\begin{aligned}
&D_1 : 729a^4 + 64b^3 = 0, b \neq 0, a \neq 0\\
&D_2 : 729a^4 + 64b^3 \neq 0, 729a^4 - 4096b^3 \neq 0, b \neq 0\\
&\qquad \text{or } b = 0, a \neq 0\\
&D_3 : 729a^4 - 4096b^3 = 0, b \neq 0, a \neq 0\\
&D_4 : a = 0, b = 0.
\end{aligned}
$$

Note that there exist inconsistent segments in SACGB's output and these segments may have common part, while all D_i's are nonempty and pairwise disjoint.

5. Discussion

This paper has presented two new modules of the `RegularChains` library for manipulating constructible sets and solving parametric polynomial systems. One of the main motivations for developing a complete collection of commands for handling constructible sets was the need of partitioning the parameter space during the computation of a comprehensive triangular decomposition. However, the `ConstructibleSetTools` module is also of great interest as an independent package. For example, as shown in the paper [5], it serves well as a *program verifier*.

Currently, some functions in the `ConstructibleSetTools` module can only accept a list of equations as input, such as `Projection` and `PolynomialMapImage`. It is desirable to remove this limitation, since, mathematically, these functions apply to any constructible set. Similarly, a comprehensive triangular decomposition for a constructible set would be also highly interesting. In the developing version of our modules, these limitations have been successfully removed and the corresponding new functions should be available in a future release.

Another ongoing project is the development of a module dedicated to *parametric semi-algebraic sets*, allowing the manipulations of parametric polynomial systems with equations, inequations and inequalities. The mathematical theory of comprehensive triangular decomposition of such sets is actually well engaged. We hope that in a near future this new module will provide a helpful support for problems in real algebraic geometry.

References

[1] P. Aubry, D. Lazard, and M. Moreno Maza. On the theories of triangular sets. *J. Symb. Comp.*, 28(1-2):105–124, 1999.

[2] E. Becker, T. Mora, M. G. Marinari, and C. Traverso. The shape of the shape lemma. In *Proc. of ISSAC*, pages 129–133, New York, NY, USA, 1994. ACM Press.

[3] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, University of Innsbruck, 1965.

[4] C. Chen, O. Golubitsky, F. Lemaire, M. Moreno Maza, and W. Pan. *Comprehensive Triangular Decomposition*, volume 4770 of *LNCS*, pages 73–101. Springer Verlag, 2007.

[5] C. Chen, M. Moreno Maza, W. Pan, and Y. Xie. On the verification of polynomial system solvers. In *Proceedings of AWFS 2007*, pages 116–144, 2007.

[6] X. Chen, P. Li, L. Lin, and D. Wang. Proving geometric theorems by partitioned-parametric Gröbner bases. In *Automated Deduction in Geometry*, pages 34–43, 2004.

[7] X. Chen and D. Wang. The projection of quasi variety and its application on geometric theorem proving and formula deduction. In *ADG*, pages 21–30, 2002.

[8] S. Chou and X. Gao. Solving parametric algebraic systems. In *Proc. ISSAC'92*, pages 335–341, 1992.

[9] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Spinger-Verlag, 1st edition, 1992.

[10] X. Dahan and É. Schost. Sharp estimates for triangular sets. In *ISSAC 04*, pages 103–110. ACM, 2004.

[11] X. Gao and D. Wang. Zero decomposition theorems for counting the number of solutions for parametric equation systems. In *Proc. ASCM 2003*, pages 129–144, 2003.

[12] L. González-Vega, C. Traverso, and A. Zanoni. Hilbert stratification and parametric Gröbner bases. In *CASC*, pages 220–235, 2005.

[13] R. D. Jenks and R. S. Sutor. *AXIOM, The Scientific Computation System*. Springer-Verlag, 1992. AXIOM is a trade mark of NAG Ltd, Oxford UK.

[14] M. Kalkbrener. A generalized euclidean algorithm for computing triangular representations of algebraic varieties. *J. Symb. Comp.*, 15:143–167, 1993.

[15] D. Kapur. An approach for solving systems of parametric polynomial equations. In V. Saraswat and P. Van Hentenryck, editors, *Principles and Practice of Constraint Programming*, pages 217–243. MIT Press, London, 1995.

[16] G. Kemper. Morphisms and constructible sets: Making two theorems of chevalley constructive. preprint, 2007.

[17] I. A. Kogan and M. Moreno Maza. Computation of canonical forms for ternary cubics. In T. Mora, editor, *Proc. ISSAC 2002*, pages 151–160. ACM Press, July 2002.

[18] D. Lazard and F. Rouillier. Solving parametric polynomial systems. *J. Symb. Comput.*, 42(6):636–667, 2007.

[19] F. Lemaire, M. Moreno Maza, and Y. Xie. The `RegularChains` library. In Ilias S. Kotsireas, editor, Maple Conference 2005, pages 355–368, 2005.

[20] F. Lemaire, M. Moreno Maza, and Y. Xie. Making a sophisticated symbolic solver available to different communities of users. In *Proc. of* Asian Technology Conference in Mathematics*'06*, 2006.

[21] X. Li and M. Moreno Maza. Multithreaded parallel implementation of arithmetic operations modulo a triangular set. In *Proc. PASCO'07*, pages 53–59, New York, NY, USA, 2006. ACM Press.

[22] X. Li, M. Moreno Maza, and E. Schost. Fast arithmetic for triangular sets: From theory to practice. In *Proc. ISSAC'07*, pages 269–276, New York, NY, USA, 2007. ACM Press.

[23] M. Manubens and A. Montes. Minimal canonical comprehensive gröber system, 01-12-06. arXiv:math.AC/0611948.

[24] M. Manubens and A. Montes. Improving the dispgb algorithm using the discriminant ideal. *J. Symb. Comput.*, 41(11):1245–1263, 2006.

[25] A. Montes. A new algorithm for discussing gröbner bases with parameters. *J. Symb. Comput.*, 33(2):183–208, 2002.

[26] M. Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report TR 4/99, NAG Ltd, Oxford, UK, 1999. http://www.csd.uwo.ca/∼moreno.

[27] J. O'Halloran and M. Schilmoeller. Gröbner bases for constructible sets. *Journal of Communications in Algebra*, 30(11), 2002.

[28] P. Schauenburg. A Gröbner-based treatment of elimination theory for affine varieties. *JSC*, 42(9):859–870, 2007.

[29] W. Sit. Computations on quasi-algebraic sets. In R. Liska, editor, *Electronic Proceedings of IMACS ACA'98*, 1998.

[30] W. Y. Sit. A theory for parametric linear systems. In *Proc. of ISSAC'91*, pages 112–121, New York, USA, 1991. ACM.

[31] A. Suzuki and Y. Sato. An alternative approach to comprehensive Gröbner bases. *JSC*, 36(3-4):649–667, 2003.

[32] A. Suzuki and Y. Sato. A simple algorithm to compute comprehensive Gröbner bases using Gröbner bases. In *ISSAC*, pages 326–331, 2006.

[33] D. Wang. Computing triangular systems and regular systems. *J. Sym. Comp.*, 30(2):221–236, 2000.

[34] D. Wang. *Elimination Methods*. Springer, 2001.

[35] D. Wang. The projection property of regular systems and its application to solving parametric polynomial systems. In A. Dolzmann, A. Seidl, and T. Sturm, editors, *Algorithmic Algebra and Logic*, pages 269–274, Herstellung und Verlag, Norderstedt, 2005.

[36] V. Weispfenning. Comprehensive Gröbner bases. *J. Symb. Comp.*, 14:1–29, 1992.

[37] V. Weispfenning. Canonical comprehensive Gröbner bases. In *ISSAC 2002*, pages 270–276. ACM Press, 2002.

[38] W. T. Wu. Basic principles of mechanical theorem proving in elementary geometries. *J. Sys. Sci. and Math. Scis*, 4:207–235, 1984.

[39] W. T. Wu. A zero structure theorem for polynomial equations solving. *MM Research Preprints*, 1:2–12, 1987.

[40] W. T. Wu. On a projection theorem of quasi-varieties in elimination theory. *Chinese Ann. Math.*, Ser. B.(11):220–226, 1990.

[41] 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.

[42] L. Yang and J. Zhang. Searching dependency between algebraic equations: an algorithm applied to automated reasoning. Technical Report IC/89/263, International Atomic Energy Agency, Miramare, Trieste, Italy, 1991.