On the Verification of Polynomial System Solvers

Changbo Chen, Marc Moreno Maza, Wei Pan and Yuzhen Xie

University of Western Ontario, London N6A 1M8, Canada

Abstract. We discuss the verification of mathematical software solving polynomial systems symbolically by way of triangular decomposition. Standard verification techniques are highly resource consuming and apply only to polynomial systems which are easy to solve. We exhibit a new approach which manipulates constructible sets represented by regular systems. We provide comparative benchmarks of different verification procedures applied to four solvers on a large set of well-known polynomial systems. Our experimental results illustrate the high efficiency of our new approach. In particular, we are able to verify triangular decompositions of polynomial systems which are not easy to solve.

Key words: Software verification, polynomial system solver, triangular decomposition.

1 Introduction

Solving systems of non-linear, algebraic or differential equations, is a fundamental problem in mathematical science. It has been studied for centuries and has stimulated many research developments. Algorithmic solutions can be classified into three categories: numeric, symbolic and hybrid numeric-symbolic. The choice for one of them depends on the characteristics of the system of equations to solve. For instance, it depends on whether the coefficients are known exactly or are approximations obtained from experimental measurements.This choice depends also on the expected answers, which could be a complete description of all the solutions, or only the real solutions, or just one sample solution among all of them.

Symbolic solvers are powerful tools in scientific computing: they are well suited for problems where the desired output must be exact and they have been applied successfully in areas like digital signal processing, robotics, theoretical physics, cryptology, dynamical systems, with many important outcomes. See [7] for an overview of these applications.

Symbolic solvers are also highly complex software. First, they implement sophisticated algorithms, which are generally at the level of on-going research. Moreover, in most computer algebra systems, the `solve` command involves nearly the entire set of libraries in the system, challenging the most advanced operations on matrices, polynomials, algebraic and modular numbers, polynomial ideals, etc.

Secondly, algorithms for solving systems of polynomial equations are by nature of exponential-space complexity. Consequently, symbolic solvers are extremely time-consuming when applied to large examples. Even worse, intermediate expressions can grow to enormous size and may halt the computations, even if the result is of moderate

size. The implementation of symbolic solvers, then, requires techniques that go far beyond the manipulation of algebraic or differential equations, such as efficient memory management, data compression, parallel and distributed computing, etc.

Last, but not least, the precise output specifications of a symbolic solver can be quite involved. Indeed, given an input polynomial system F, defining what a symbolic solver should return implies describing what the geometry of the solution set $\mathbf{V}(F)$ of F can be. For an arbitrary F, the set $\mathbf{V}(F)$ may consist of components of different natures and sizes: points, lines, curves, surfaces. This leads to the following difficult challenge.

Given a polynomial system F and a set of components $C_1, \ldots, C_e$, it is hard, in general, to tell whether the union of $C_1, \ldots, C_e$ corresponds exactly to the solution set $\mathbf{V}(F)$ or not. Actually, solving this verification problem is generally (at least) as hard as solving the system F itself.

Because of the high complexity of symbolic solvers, developing verification algorithms and reliable verification software tools is a clear need. However, this verification problem has received little attention in the literature. In this paper, we present new techniques for verifying a large class of symbolic solvers. We also report on intensive experimentation illustrating the high efficiency of our approach w.r.t. known techniques.

We assume that each component of the solution set $\mathbf{V}(F)$ is given by a so-called *regular system*. This is a natural assumption in symbolic computations, well-developed in the literature under different terminologies, see [1, 22] and the references therein. In broad words, a regular system consists of several polynomial equations with a triangular shape

$$p_1(x_1) = p_2(x_1, x_2) = \cdots = p_i(x_1, x_2, \ldots, x_n) = 0$$

and a polynomial inequality

$$h(x_1, \ldots, x_n) \neq 0$$

such that there exists (at least) one point $(a_1, \ldots, a_n)$ satisfying the above equations and inequality. Note that these polynomials may contain parameters.

Let us consider the following well-known system F taken from [5].

$$\begin{cases} x^{31} - x^6 - x - y = 0 \\ x^8 - z = 0 \\ x^{10} - t = 0 \end{cases}$$

We aim at solving this system for $x > y > z > t$, that is, expressing x as a function of y, z, t, then y as a function of z, t and z as a function of t. One possible decomposition is given by the three regular systems below:

$$\begin{cases} \left(t^4 - t\right) x - ty - z^2 = 0 \\ t^3 y^2 + 2t^2 z^2 y + \left(-t^6 + 2t^3 + t - 1\right) z^4 = 0 \\ z^5 - t^4 = 0 \\ t^4 - t \neq 0 \end{cases}, \quad \begin{cases} x^2 - z^4 = 0 \\ y + t^2 z^2 = 0 \\ z^5 - t = 0 \\ t^3 - 1 = 0 \end{cases}, \quad \begin{cases} x = 0 \\ y = 0 \\ z = 0 \\ t = 0 \end{cases}$$

Another decomposition is given by these other three regular systems:

$$\left\{\begin{array}{r} \left(t^4-t\right)x - ty - z^2 = 0 \\ tzy^2 + 2z^3y - t^8 + 2t^5 + t^3 - t^2 = 0 \\ z^5 - t^4 = 0 \\ z\left(t^4-t\right) \neq 0 \end{array}\right., \quad \left\{\begin{array}{r} zx^2 - t = 0 \\ ty + z^2 = 0 \\ z^5 - t = 0 \\ t^3 - 1 = 0 \\ tz \neq 0 \end{array}\right., \quad \left\{\begin{array}{r} x = 0 \\ y = 0 \\ z = 0 \\ t = 0 \end{array}\right.$$

These two decompositions look slightly different (in particular, the second components) and one could think that, if each of them was produced by a different solver, then at least one of these solvers has a bug. In fact, both decompositions are valid, but proving respectively that they encode the solution set $\mathbf{V}(F)$ is not feasible without computer assistance. However, proving that they define the same set of points can be achieved by an expert hand without computer assistance. This is an important observation that we will guide us in this paper.

Let us consider now an arbitrary input system F and a set of components $C_1, \ldots, C_e$ encoded by regular systems $S_1, \ldots, S_e$ respectively. The usual approach for verifying that $C_1, \ldots, C_e$ correspond exactly to the solution set $\mathbf{V}(F)$ is as follows.

(1) First, one checks that each candidate component C_i is actually contained in $\mathbf{V}(F)$. This essentially reduces to substitute the coordinates of the points given by C_i into the polynomials of F: if all these polynomials vanish at these points, then C_i is a component of $\mathbf{V}(F)$, otherwise, (and up to technical details that we will skip in this introduction) C_i is not a component of $\mathbf{V}(F)$.
(2) Secondly, one checks that $\mathbf{V}(F)$ is contained in the union of the candidate components $C_1, \ldots, C_e$ by:
  (2.1) computing a polynomial system G such that $\mathbf{V}(G)$ corresponds exactly to $C_1, \ldots, C_e$, and
  (2.2) checking that every solution of $\mathbf{V}(F)$ cancels the polynomials of G.

Steps (2.1) and (2.2) can be performed using standard techniques based on computations of Gröbner bases, as we discuss in Section 6.1. These calculations are very expensive, as shown by our experimentation, reported in Section 7.

In this paper, we propose a different approach, summarized in non-technical language in Section 2. The main idea is as follows. Instead of comparing a candidate set of components $C_1, \ldots, C_e$ against the input system F, we compare it against the output $D_1, \ldots, D_f$ produced by another solver. Both this solver and the comparison process are assumed to be validated. Hence, the candidate set of components $C_1, \ldots, C_e$ corresponds exactly to the solution set $\mathbf{V}(F)$ if and only if the comparison process shows that $D_1, \ldots, D_f$ and $C_1, \ldots, C_e$ define the same solution set.

The technical details of this new approach are given in Sections 3, 4, 5 and 6. In Section 3, we review the fundamental algebraic concepts and operations involved in our work. In particular, we specify the kind of solvers that we consider in this study, namely those solving polynomial systems by means of *triangular decompositions*.

The key computational concept behind these triangular decomposition computed is that of a *constructible set*, so we dedicate Section 4 to it. Section 5 is a formal and complete presentation of our process for comparing triangular decompositions. In Section 6,

we summarize the different verification procedures that are available for triangular decompositions, including our new approach. In Section 7, we report on experimentation with these verification procedures. Our data illustrate the high efficiency of our new approach.

2 Methodology

Let us consider again an arbitrary input polynomial system F and a set of components $C_1, \ldots, C_e$ encoded by regular systems $S_1, \ldots, S_e$ respectively. As mentioned in the Introduction, checking whether $C_1, \ldots, C_e$ corresponds exactly to the solution set $\mathbf{V}(F)$ of F can be done by means of Gröbner bases computations. This verification process is quite simple, see Section 6, and its implementation is straightforward, Thus, if the underlying Gröbner bases engine is *reliable*, such verification tool can be regarded as safe. See [2] for details.

Unfortunately, this verification process is highly expensive. Even worse, as shown by our experimental results in Section 7, this verification process is unable to check many triangular decompositions that are easy to compute.

We propose a new approach in order to overcome this limitation. Assume that we have at hand a reliable solver computing triangular decompositions of polynomial systems. We believe that this reliability can be acquired over time by combining several features.

- Checking the solver with a verification tool based on Gröbner bases for input systems of moderate difficulty.
- Using the solver for input systems of higher difficulty where the output can be verified by theoretical arguments, see [3] for an example of such input system.
- Involving the library supporting the solver in other applications.
- Making the solver widely available to potential users.

Suppose that we are currently developing a new solver computing triangular decompositions. In order to verify the output of this new solver, we can take advantage of the reliable solver.

This may sound natural and easy in the first place, but this is actually not. Indeed, as shown in the Introduction, two different solvers can produce two different, but valid, triangular decompositions for the same input system. Checking that these two triangular decompositions encode the same solution set boils down to compute the differences of two constructible sets. This is a non-trivial operation, see the survey paper [16].

The first contribution of our paper is to provide a relatively simple, but efficient, procedure for computing the set theoretical differences between two constructible sets. See Section 5. Such procedure can be used to develop a verification tool for our new solver by means of our reliable solver. Moreover, this procedure is sufficiently straightforward to implement such that it can be trusted after a relatively short period of testing, as the case for the verification tool based on Gröbner bases computations.

The second contribution of our work is to illustrate the high efficiency of this new verification tool. In Section 7, we consider four solvers computing triangular decomposition of polynomial systems:

– the command *Triangularize* of the *RegularChains* library [11] in MAPLE
– the TRIADE solver of the *BasicMath* library [8] in ALDOR
– the commands *RegSer* and *SimSer* of the *Epsilon* library [19] in MAPLE.

We have run these four solvers on a large set of well-known input systems from the data base [12, 17, 21]. For those systems for which this is feasible, we have verified their computed triangular decompositions with a verification tool based on Gröbner bases computations. Then, for each input system, we have compared all its computed triangular decompositions by means of our new verification tool.

Based on our experimentation data reported in Section 7 we make the following observations.

– All computed triangular decompositions, that could be checked via Gröbner bases computations, are correct.
– However, the verification tool based on Gröbner bases computations failed to check many examples by running out of computer memory.
– For each input system F, most pairs of triangular decompositions of F could be compared successfully by our new verification tool.
– Moreover, for any system F to which all verification tools could be applied, our new approach runs much faster.

This suggests that our four solvers and our new verification tool have a good level of reliability. Moreover, our verification tool allows to process cases that were previously out of reach.

3 Preliminaries

In this section we introduce notations and review fundamental results in the theory of regular chains and regular systems [1, 4, 10, 13, 18, 21]. We shall use some notions from commutative algebra (such as the dimension of an ideal) and refer for instance to [15] for this subject.

3.1 Basic notations and definitions

Let $\mathbb{K}[Y] := \mathbb{K}[Y_1, \ldots, Y_n]$ be the polynomial ring over the field $\mathbb{K}$ in variables $Y_1 < \cdots < Y_n$. Let $p \in \mathbb{K}[Y]$ be a non-constant polynomial. The *leading coefficient* and the *degree* of p regarded as a univariate polynomial in Y_i will be denoted by $\mathrm{lc}(p, Y_i)$ and $\deg(p, Y_i)$ respectively. The greatest variable appearing in p is called the *main variable* denoted by $\mathrm{mvar}(p)$. The *separant* $\mathrm{sep}(p)$ of p w.r.t $\mathrm{mvar}(p)$, is $\partial p / \partial \mathrm{mvar}(p)$. The degree, the leading coefficient, and the leading monomial of p regarding as a univariate polynomial in $\mathrm{mvar}(p)$ are called the *main variable*, the *initial*, and the *rank* of p; they are denoted by $\mathrm{mdeg}(p)$, $\mathrm{init}(p)$ and $\mathrm{rank}(p)$ respectively.

Let $F \subset \mathbb{K}[Y]$ be a finite polynomial set. Denote by $\langle F \rangle$ the ideal it generates in $\mathbb{K}[Y]$ and by $\sqrt{\langle F \rangle}$ the radical of $\langle F \rangle$. Let h be a polynomial in $\mathbb{K}[Y]$, the *saturated ideal* $\langle F \rangle : h^\infty$ of $\langle F \rangle$ w.r.t h, is the set

$$\{q \in \mathbb{K}[Y] \mid \exists m \in \mathbb{N} \text{ s.t. } h^m q \in \langle F \rangle\},$$

which is an ideal in $\mathbb{K}[Y]$.

A polynomial $p \in \mathbb{K}[Y]$ is a *zerodivisor* modulo $\langle F \rangle$ if there exists a polynomial q such that pq is zero modulo $\langle F \rangle$, and q is not zero modulo $\langle F \rangle$. The polynomial is *regular* modulo $\langle F \rangle$ if it is neither zero, nor a zerodivisor modulo $\langle F \rangle$. Denote by $\mathbf{V}(F)$ the *zero set* (or solution set, or algebraic variety) of F in $\overline{\mathbb{K}}^n$. For a subset $W \subset \overline{\mathbb{K}}^n$, denote by $\overline{W}$ its closure in the Zariski topology, that is the intersection of all algebraic varieties $\mathbf{V}(G)$ containing W for all $G \subset \mathbb{K}[Y]$.

Let $T \subset \mathbb{K}[Y]$ be a *triangular set*, that is a set of non-constant polynomials with pairwise distinct main variables. Denote by $\text{mvar}(T)$ the set of main variables of $t \in T$. A variable in Y is called *algebraic* w.r.t. T if it belongs to $\text{mvar}(T)$, otherwise it is called *free* w.r.t. T. For a variable $v \in Y$ we denote by $T_{<v}$ (resp. $T_{>v}$) the subsets of T consisting of the polynomials t with main variable less than (resp. greater than) v. If $v \in \text{mvar}(T)$, we say T_v is defined. Moreover, we denote by T_v the polynomial in T whose main variable is v, by $T_{\leqslant v}$ the set of polynomials in T with main variables less than or equal to v and by $T_{\geqslant v}$ the set of polynomials in T with main variables greater than or equal to v.

Definition 1 *Let $p, q \in \mathbb{K}[Y]$ be two nonconstant polynomials. We say* $\text{rank}(p)$ *is smaller than* $\text{rank}(q)$ *w.r.t* Ritt ordering *and we write,* $\text{rank}(p) <_r \text{rank}(q)$ *if one of the following assertions holds:*

- $\text{mvar}(p) < \text{mvar}(q)$,
- $\text{mvar}(p) = \text{mvar}(q)$ *and* $\text{mdeg}(p) < \text{mdeg}(q)$.

Note that the partial order $<_r$ is a well ordering. Let $T \subset \mathbb{K}[Y]$ be a triangular set. Denote by $\text{rank}(T)$ the set of $\text{rank}(p)$ for all $p \in T$. Observe that any two ranks in $\text{rank}(T)$ are comparable by $<_r$. Given another triangular set $S \subset \mathbb{K}[Y]$, with $\text{rank}(S) \neq \text{rank}(T)$, we write $\text{rank}(T) <_r \text{rank}(S)$ whenever the minimal element of the symmetric difference $(\text{rank}(T) \setminus \text{rank}(S)) \cup (\text{rank}(S) \setminus \text{rank}(T))$ belongs to $\text{rank}(T)$. By $\text{rank}(T) \leqslant_r \text{rank}(S)$, we mean either $\text{rank}(T) < \text{rank}(S)$ or $\text{rank}(T) = \text{rank}(S)$. Note that any sequence of triangular sets, of which ranks strictly decrease w.r.t $<_r$, is finite.

Given a triangular set $T \subset \mathbb{K}[Y]$, denote by h_T be the product of the initials of T (throughout the paper we use this convention and when T consists of a single element g we write it in h_g for short). The *quasi-component* $\mathbf{W}(T)$ of T is $\mathbf{V}(T) \setminus \mathbf{V}(h_T)$, in other words, the points of $\mathbf{V}(T)$ which do not cancel any of the initials of T. We denote by $\text{Sat}(T)$ the *saturated ideal of* T: if T is empty then $\text{Sat}(T)$ is defined as the trivial ideal $\langle 0 \rangle$, otherwise it is the ideal $\langle T \rangle : h_T^\infty$.

Let $h \in \mathbb{K}[Y]$ be a polynomial and $F \subset \mathbb{K}[Y]$ a set of polynomials, we write

$$\mathbf{Z}(F, T, h) := (\mathbf{V}(F) \cap \mathbf{W}(T)) \setminus \mathbf{V}(h).$$

When F consists of a single polynomial p, we use $\mathbf{Z}(p, T, h)$ instead of $\mathbf{Z}(\{p\}, T, h)$; when F is empty we just write $\mathbf{Z}(T, h)$. By $\mathbf{Z}(F, T)$, we denote $\mathbf{V}(F) \cap \mathbf{W}(T)$.

Given a family of pairs $\mathbf{S} = \{[T_i, h_i] \mid 1 \leq i \leq e\}$, where $T_i \subset \mathbb{K}[Y]$ is a triangular set and $h_i \in \mathbb{K}[Y]$ is a polynomial. We write

$$\mathbf{Z}(S) := \bigcup_{i=1}^{e} \mathbf{Z}(T_i, h_i).$$

We conclude this section with some well known properties of ideals and triangular sets. For a proper ideal $\mathcal{I}$, we denote by $\dim(\mathbf{V}(\mathcal{I}))$ the dimension of $\mathbf{V}(\mathcal{I})$.

Lemma 1 *Let $\mathcal{I}$ be a proper ideal in $\mathbb{K}[Y]$ and $p \in \mathbb{K}[Y]$ be a polynomial regular w.r.t $\mathcal{I}$. Then, either $\mathbf{V}(\mathcal{I}) \cap \mathbf{V}(p)$ is empty or we have: $\dim(\mathbf{V}(\mathcal{I}) \cap \mathbf{V}(p)) \leq \dim(\mathbf{V}(\mathcal{I})) - 1$.*

Lemma 2 *Let T be a triangular set in $\mathbb{K}[Y]$. Then, we have*

$$\overline{\mathbf{W}(T)} \setminus \mathbf{V}(h_T) = \mathbf{W}(T) \text{ and } \overline{\mathbf{W}(T)} \setminus \mathbf{W}(T) = \mathbf{V}(h_T) \cap \overline{\mathbf{W}(T)}.$$

PROOF. Since $\mathbf{W}(T) \subseteq \overline{\mathbf{W}(T)}$, we have

$$\mathbf{W}(T) = \mathbf{W}(T) \setminus \mathbf{V}(h_T) \subseteq \overline{\mathbf{W}(T)} \setminus \mathbf{V}(h_T).$$

On the other hand, $\overline{\mathbf{W}(T)} \subseteq \mathbf{V}(T)$ implies

$$\overline{\mathbf{W}(T)} \setminus \mathbf{V}(h_T) \subseteq \mathbf{V}(T) \setminus \mathbf{V}(h_T) = \mathbf{W}(T).$$

This proves the first claim. Observe that we have:

$$\overline{\mathbf{W}(T)} = \left(\overline{\mathbf{W}(T)} \setminus \mathbf{V}(h_T)\right) \cup \left(\overline{\mathbf{W}(T)} \cap \mathbf{V}(h_T)\right).$$

We deduce the second one.

Lemma 3 ([1, 4]) *Let T be a triangular set in $\mathbb{K}[Y]$. Then, we have*

$$\mathbf{V}(\text{Sat}(T)) = \overline{\mathbf{W}(T)}.$$

Assume furthermore that $\mathbf{W}(T) \neq \emptyset$ holds. Then $\mathbf{V}(\text{Sat}(T))$ is a nonempty unmixed algebraic set with dimension $n - |T|$. Moreover, if N is the free variables of T, then for every prime ideal $\mathcal{P}$ associated with Sat(T) *we have*

$$\mathcal{P} \cap \text{Sat}(T) = \langle 0 \rangle.$$

3.2 Regular chain and regular system

Definition 2 (Regular Chain) *A triangular set $T \subset \mathbb{K}[Y]$ is a* regular chain *if one of the following conditions hold:*

- *either T is empty,*
- *or $T \setminus \{T_{\max}\}$ is a regular chain, where $T_{\max}$ is the polynomial in T with maximum rank, and the initial of $T_{\max}$ is regular w.r.t.* Sat$(T \setminus \{T_{\max}\})$.

To deal with inequalities, one can introduce the notion of a regular system via that of a regular chain.

Definition 3 (Regular System) *A pair $[T, h]$ is a regular system if T is a regular chain, and $h \in \mathbb{K}[Y]$ is regular w.r.t* Sat(T).

Remark 1 *A stronger notion of a regular system was presented in [18]. For example, the polynomial system $[T, h]$, with $T = [Y_1Y_4 - Y_2]$ and $h = Y_2Y_3$, is still a regular system in our sense but not in that of [18]. Our definition is more convenient for our purpose in dealing with zerodivisors. We also note that in dimension zero (no free variables exist) the notion of a regular chain (as used in this paper) and that of a regular set introduced in [18] coincide, see [1, 18] for details.*

Proposition 1 *For every regular system $[T, h]$ we have $\mathbf{Z}(T, h) \neq \emptyset$.*

PROOF. Since T is a regular chain, by Lemma 3 we have $\mathbf{V}(\text{Sat}(T)) \neq \emptyset$. By definition of regular system, the polynomial hh_T is regular w.r.t Sat(T). Hence, by Lemma 1, the set $\mathbf{V}(hh_T) \cap \mathbf{V}(\text{Sat}(T))$ either is empty, or has lower dimension than $\mathbf{V}(\text{Sat}(T))$. Therefore, the set

$$\mathbf{V}(\text{Sat}(T)) \setminus \mathbf{V}(hh_T) = \mathbf{V}(\text{Sat}(T)) \setminus (\mathbf{V}(hh_T) \cap \mathbf{V}(\text{Sat}(T)))$$

is not empty. Finally, by Lemma 2, the set

$$\mathbf{Z}(T, h) = \mathbf{W}(T) \setminus \mathbf{V}(h) = \overline{\mathbf{W}(T)} \setminus \mathbf{V}(hh_T) = \mathbf{V}(\text{Sat}(T)) \setminus \mathbf{V}(hh_T)$$

is not empty.

Notation 1 *For a regular system $R = [T, h]$, we define* rank$(R) :=$ rank(T). *For a set $\mathcal{R}$ of regular systems, we define*

$$\text{rank}(\mathcal{R}) := \max\{\text{rank}(T) \mid [T, h] \in \mathcal{R}\}.$$

For a pair of regular systems (L, R), we define rank$((L, R)) :=$ (rank(L), rank(R)). *For a pair of lists of regular systems, we define*

$$\text{rank}((\mathcal{L}, \mathcal{R})) = (\text{rank}(\mathcal{L}), \text{rank}(\mathcal{R})).$$

For triangular sets $T, T_1, \ldots, T_e$ we write $\mathbf{W}(T) \xrightarrow{D} (\mathbf{W}(T_i), i = 1 \ldots e)$ if one of the following conditions holds:

- *either $e = 1$ and $T = T_1$,*
- *or $e > 1$,* rank$(T_i) <$ rank(T) *for all $i = 1 \ldots e$ and*

$$\mathbf{W}(T) \subseteq \bigcup_{i=1}^{e} \mathbf{W}(T_i) \subseteq \overline{\mathbf{W}(T)}.$$

3.3 Triangular decompositions

Definition 4 *Given a finite polynomial set $F \subset \mathbb{K}[Y]$, a* triangular decomposition *of $\mathbf{V}(F)$ is a finite family $\mathcal{T}$ of regular chains of $\mathbb{K}[Y]$ such that*

$$\mathbf{V}(F) = \bigcup\nolimits_{T \in \mathcal{T}} \mathbf{W}(T).$$

For a finite polynomial set $F \subset \mathbb{K}[Y]$, the TRIADE algorithm [13] computes a triangular decomposition of $\mathbf{V}(F)$. We list below the specifications of the operations from TRIADE that we use in this paper.

Let p, p_1, p_2 be polynomials, and let T, C, E be regular chains such that $C \cup E$ is a triangular set (but not necessarily a regular chain).

- $\mathbf{Regularize}(p, T)$ returns regular chains $T_1, \ldots, T_e$ such that
  - $\mathbf{W}(T) \xrightarrow{D} (\mathbf{W}(T_i), i = 1 \ldots e)$,
  - for all $1 \leq i \leq e$ the polynomial p is either 0 or regular modulo $\mathrm{Sat}(T_i)$.
- For a set of polynomials F, $\mathbf{Triangularize}(F, T)$ returns regular chains $T_1, \ldots, T_e$ such that we have

$$\mathbf{V}(F) \cap \mathbf{W}(T) \ \subseteq \ \mathbf{W}(T_1) \cup \cdots \cup \mathbf{W}(T_e) \ \subseteq \ \mathbf{V}(F) \cap \overline{\mathbf{W}(T)}.$$

  and for $1 \leq i \leq e$ we have $\mathrm{rank}(T_i) < \mathrm{rank}(T)$ whenever $F \not\subset \mathrm{Sat}(T)$.
- $\mathbf{Extend}(C \cup E)$ returns a set of regular chains $\{C_i \mid i = 1 \ldots e\}$ such that we have $\mathbf{W}(C \cup T) \xrightarrow{D} (\mathbf{W}(C_i), i = 1 \ldots e)$.
- Assume that p_1 and p_2 are two non-constant polynomials with the same main variable v, which is larger than any variable appearing in T, and assume that the initials of p_1 and p_2 are both regular w.r.t. $\mathrm{Sat}(T)$. Then, $\mathbf{GCD}(p_1, p_2, T)$ returns a sequence

$$([g_1, C_1], \ldots, [g_d, C_d], [\emptyset, D_1], \ldots, [\emptyset, D_e]),$$

  where g_i are polynomials and C_i, D_i are regular chains such that the following properties hold:
  - $\mathbf{W}(T) \xrightarrow{D} (\mathbf{W}(C_1), \ldots, \mathbf{W}(C_d), \mathbf{W}(D_1), \ldots, \mathbf{W}(D_e))$,
  - $\dim \mathbf{V}(\mathrm{Sat}(C_i)) = \dim \mathbf{V}(\mathrm{Sat}(T))$ and $\dim \mathbf{V}(\mathrm{Sat}(D_j)) < \dim \mathbf{V}(\mathrm{Sat}(T))$, for all $1 \leqslant i \leqslant d$ and $1 \leqslant j \leqslant e$,
  - the leading coefficient of g_i w.r.t. v is regular w.r.t. $\mathrm{Sat}(C_i)$,
  - for all $1 \leqslant i \leqslant d$ there exists polynomials u_i and v_i such that we have $g_i = u_i p_1 + v_i p_2 \mod \mathrm{Sat}(C_i)$,
  - if g_i is not constant and its main variable is v, then p_1 and p_2 belong to $\mathrm{Sat}(C_i \cup \{g_i\})$.

4 Representations of constructible sets

Constructible set [6, 9] is a classical concept in elimination theory. In this section, we present two types of representations for constructible sets in $\overline{\mathbb{K}}^n$.

Definition 1 (Constructible set). *A constructible subset of $\overline{\mathbb{K}}^n$ is any 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 over $\mathbb{K}$.

Let $\mathcal{F}$ be the set of all constructible subsets of $\overline{\mathbb{K}}^n$ w.r.t $\mathbb{K}$. From Exercise 3.18 in [9], we have

– all open algebraic sets are in $\mathcal{F}$;
– the complement of an element in $\mathcal{F}$ is in $\mathcal{F}$;
– the intersection of two elements in $\mathcal{F}$ is in $\mathcal{F}$.

Moreover, these three properties describe *exactly* all constructible sets. Given a set of polynomial F and $f \in \mathbb{K}[Y]$, we denote $\mathbf{D}(F, f)$ the difference of $\mathbf{V}(F) \setminus \mathbf{V}(f)$, which is also called a *basic constructible set*. If F is the empty set, then we write $\mathbf{D}(f)$ for short. Note that for a regular system in [20], we have $\mathbf{D}(T, h) = \mathbf{Z}(T, h)$.

4.1 Gröbner basis representation

Now Gröbner bases have become a standard tool to deal with algebraic sets; and they can be applied to manipulate constructible sets as well. Given a constructible set C, according to the definition, one can represent C by a unique sequence of closed algebraic sets whose defining ideals naturally can be characterized by their reduced Gröbner bases [14].

However, the constructible sets are intrinsically geometrical objects. We pay extra cost to manipulate them, since it is very hard to compute the intersection of two ideals and even to compute the radical ideal of an ideal. Whatsoever, there exist effective algorithms to manipulate constructible sets. We shall use regular systems to do the same jobs in a more efficient manner.

4.2 Regular system representation

In this section, we show that (Theorem 2) every constructible set C can be represented by a finite set of regular systems$\{[T_i, h_i] \mid i = 1 \ldots e\}$, that is,

$$C = \bigcup_{i=1}^{e} \mathbf{Z}(T_i, h_i).$$

Combining with Lemma 1, we know that if a regular system representation of a constructible set is empty, then the constructible set C is empty. This fact leads to an important application of verifying polynomial system solvers. The proof of Theorem 2 is constructive and relies on algorithms called $\mathbf{Difference}$ and $\mathbf{DifferenceLR}$, presented in Section 5. As an immediate consequence of the specifications of these algorithms formally proved in the next section, we obtain the following theorem.

Theorem 1. *Given two regular systems $[T, h]$ and $[T', h']$, there is an algorithm to compute the regular system representations of:*

(1) *the difference* $\mathbf{Z}(T, h) \setminus \mathbf{Z}(T', h')$*;*
(2) *the intersection* $\mathbf{Z}(T, h) \bigcap \mathbf{Z}(T', h')$.

PROOF. Clain (1) follows from the specifications of the $\mathbf{Difference}$ algorithm. Claim (2) follows from (1), together with the fact that for any two sets A and B, we have $A \cap B = A \setminus (A \setminus B)$.

Theorem 2. *Every constructible set can be represented by a finite set of regular systems.*

PROOF. Consider the following family $\widetilde{\mathcal{F}}$ of subsets of $\overline{\mathbb{K}}^n$:

$$\widetilde{\mathcal{F}} = \{S \mid S = \bigcup_{i=1}^{e} \mathbf{Z}(T_i, p_i)\},$$

where $[T_i, p_i]$ are regular systems. First, every open subset can be decomposed into a finite union of open subsets $\mathbf{D}(f)$, where f is a polynomial. Each $\mathbf{D}(f)$ can be represented by the regular system $[\emptyset, f]$ consisting of the empty regular chain and f. Hence $\widetilde{\mathcal{F}}$ contains all open subsets. Secondly, consider two elements S and T in $\widetilde{\mathcal{F}}$; and assume that

$$S = \bigcup_{i=1}^{e} \mathbf{Z}(S_i, p_i) \quad \text{and} \quad T = \bigcup_{j=1}^{f} \mathbf{Z}(T_j, q_j).$$

We have

$$S \bigcap T = \bigcup_{i=1}^{e} \bigcup_{j=1}^{f} \left(\mathbf{Z}(S_i, p_i) \bigcap \mathbf{Z}(T_j, q_j) \right).$$

By Theorem 1, $S \bigcap T$ has a regular system representation, that is to say, $S \bigcap T \in \widetilde{\mathcal{F}}$. By induction, any finite intersection of elements of $\widetilde{\mathcal{F}}$ is in $\widetilde{\mathcal{F}}$. Finally, we shall prove that the complement of an element in $\widetilde{\mathcal{F}}$ is in $\widetilde{\mathcal{F}}$. Essentially, we only need to show that for each $1 \leq i \leq e$, $\mathbf{Z}(S_i, p_i)^c$ is in $\widetilde{\mathcal{F}}$. Indeed,

$$\mathbf{Z}(S_i, p_i)^c = \mathbf{W}(S_i)^c \bigcup \mathbf{V}(p_i) = \mathbf{V}(S_i)^c \bigcup \mathbf{V}(p_i h_{S_i})$$

is in $\widetilde{\mathcal{F}}$, since both $\mathbf{V}(S_i)^c$ and $\mathbf{V}(p_i h_{S_i})$ have regular system representations.

5 The Difference algorithms

In this section, we present an algorithm to compute the set theoretical difference of two constructible sets given by regular systems. Two procedures are, actually involved, in order to achieve this goal, **Difference** and **DifferenceLR**. Their specifications and pseudo-codes can be found below. The rest of this section is dedicated to proving the correctness and termination of these algorithms. For the pseudo-code, we use the MAPLE syntax. However, each of the two functions below returns a sequence of values. Individual value or sub-sequences of the returned sequence are thrown to the flow of output by means of an **output** statement. Hence an **output** statement does not cause the termination of the function execution.

Algorithm 1 **Difference**$([T, h], [T', h'])$
Input $[T, h]$, $[T', h']$ two regular systems.

Output Regular systems $\{[T_i, h_i] \mid i = 1 \ldots e\}$ such that

$$\mathbf{Z}(T, h) \setminus \mathbf{Z}(T', h') = \bigcup_{i=1}^{e} \mathbf{Z}(T_i, h_i),$$

and $\text{rank}(T_i) \leqslant_r \text{rank}(T)$.

Algorithm 2 $\mathbf{DifferenceLR}(\mathcal{L}, \mathcal{R})$

Input $\mathcal{L} := \{[L_i, f_i] \mid i = 1 \ldots r\}$ and $\mathcal{R} := \{[R_j, g_j] \mid j = 1 \ldots s\}$ two lists of regular systems,

Output Regular systems $\mathcal{S} := \{[T_i, h_i] \mid i = 1 \ldots e\}$ such that

$$\left(\bigcup_{i=1}^{r} \mathbf{Z}(L_i, f_i)\right) \setminus \left(\bigcup_{j=1}^{s} \mathbf{Z}(R_j, g_j)\right) = \bigcup_{i=1}^{e} \mathbf{Z}(T_i, h_i),$$

with $\text{rank}(\mathcal{S}) \leqslant_r \text{rank}(\mathcal{L})$

To prove the termination and correctness of above two algorithms, we present a series of technical lemmas.

Lemma 4 *Let p and h be polynomials and T a regular chain. Assume that $p \notin \text{Sat}(T)$. Then there exists an operation $\mathbf{Intersect}(p, T, h)$ returning a set of regular chains $\{T_1, \ldots, T_e\}$ such that*

(i) *h is regular w.r.t $\text{Sat}(T_i)$ for all i;*
(ii) *$\text{rank}(T_i) <_r \text{rank}(T)$;*
(iii) *$\mathbf{Z}(p, T, h) \subseteq \cup_{i=1}^{e} \mathbf{Z}(T_i, h) \subseteq (\mathbf{V}(p) \cap \overline{\mathbf{W}(T)}) \setminus \mathbf{V}(h)$;*
(iv) *Moreover, if the product of initials h_T of T divides h then*

$$\mathbf{Z}(p, T, h) = \bigcup_{i=1}^{e} \mathbf{Z}(T_i, h).$$

PROOF. Let

$$\mathcal{S} = \mathbf{Triangularize}(p, T),$$
$$\mathcal{R} = \bigcup_{C \in \mathcal{S}} \mathbf{Regularize}(h, C).$$

We then have

$$\mathbf{V}(p) \cap \mathbf{W}(T) \subseteq \bigcup_{R \in \mathcal{R}} \subseteq \mathbf{V}(p) \cap \overline{\mathbf{W}(T)}.$$

This implies

$$\mathbf{Z}(p, T, h) \subseteq \bigcup_{R \in \mathcal{R},\ h \notin \text{Sat}(R)} \mathbf{Z}(R, h) \subseteq (\mathbf{V}(p) \cap \overline{\mathbf{W}(T)}) \setminus \mathbf{V}(h).$$

Rename the regular chains $\{R \mid R \in \mathcal{R},\ h \notin \text{Sat}(R)\}$ as $\{T_1, \ldots, T_e\}$. By the specification of $\mathbf{Regularize}$ we immediately conclude $(i), (iii)$ hold. Since $h \notin \text{Sat}(T)$, by the specialization of $\mathbf{Triangularize}$, (ii) holds. By Lemma 2, (iv) holds.

Algorithm 1 $\mathbf{Difference}([T, h], [T', h'])$

if $\text{Sat}(T) = \text{Sat}(T')$ **then**
  output $\mathbf{Intersect}(h'h_{T'}, T, hh_T)$
else
  Let v be the largest variable s.t. $\text{Sat}(T_{<v}) = \text{Sat}(T'_{<v})$
  if $v \in \text{mvar}(T')$ and $v \notin \text{mvar}(T)$ **then**
    $p' \leftarrow T'_v$
    output $[T, hp']$
    output $\mathbf{DifferenceLR}(\mathbf{Intersect}(p', T, hh_T), [T', h'])$
  else if $v \notin \text{mvar}(T')$ and $v \in \text{mvar}(T)$ **then**
    $p \leftarrow T_v$
    output $\mathbf{DifferenceLR}([T, h], \mathbf{Intersect}(p, T', h'h_{T'}))$
  else
    $p \leftarrow T_v$
    $\mathcal{G} \leftarrow \mathbf{GCD}(T_v, T'_v, T_{<v})$
    if $|\mathcal{G}| = 1$ **then**
      Let $(g, C) \in \mathcal{G}$
      if $g \in \mathbb{K}$ **then**
        output $[T, h]$
      else if $\text{mvar}(g) < v$ **then**
        output $[T, gh]$
        output $\mathbf{DifferenceLR}(\mathbf{Intersect}(g, T, hh_T), [T', h'])$
      else if $\text{mvar}(g) = v$ **then**
        if $\text{mdeg}(g) = \text{mdeg}(p)$ **then**
          $D'_p \leftarrow T'_{<v} \cup \{p\} \cup T'_{>v}$
          output $\mathbf{Difference}([T, h], [D'_p, h'h_{T'}])$
        else if $\text{mdeg}(g) < \text{mdeg}(p)$ **then**
          $q \leftarrow \text{pquo}(p, g, C)$
          $D_g \leftarrow C \cup \{g\} \cup T_{>v}$
          $D_q \leftarrow C \cup \{q\} \cup T_{>v}$
          output $\mathbf{Difference}([D_g, hh_T], [T', h'])$
          output $\mathbf{Difference}([D_q, hh_T], [T', h'])$
          output $\mathbf{DifferenceLR}(\mathbf{Intersect}(h_g, T, hh_T), [T', h'])$
        end if
      end if
    else if $|\mathcal{G}| \geq 2$ **then**
      for $(g, C) \in \mathcal{G}$ **do**
        if $|C| > |T_{<v}|$ **then**
          for $E \in \mathbf{Extend}(C, T_{\geqslant v})$ **do**
            for $D \in \mathbf{Regularize}(hh_T, E)$ **do**
              if $hh_T \notin \text{Sat}(D)$ **then**
                output $\mathbf{Difference}([D, hh_T], [T', h'])$
              end if
            end for
          end for
        else
          output $\mathbf{Difference}([C \cup T_{\geqslant v}, hh_T], [T', h'])$
        end if
      end for
    end if
  end if
end if

Algorithm 2 $\mathbf{DifferenceLR}(L, R)$

if $L = \emptyset$ **then**
  output $\emptyset$
else if $R = \emptyset$ **then**
  output L
else if $|R| = 1$ **then**
  Let $[T', h'] \in R$
  for $[T, h] \in L$ **do**
    output $\mathbf{Difference}([T, h], [T', h'])$
  end for
else
  while $R \neq \emptyset$ **do**
    Let $[T', h'] \in R$, $R \leftarrow R \setminus \{ [T', h'] \}$
    $S \leftarrow \emptyset$
    for $[T, h] \in L$ **do**
      $S \leftarrow S \cup \mathbf{Difference}([T, h], [T', h'])$
    end for
    $L \leftarrow S$
  end while
end if

Lemma 5 *Let* $[T, h]$ *and* $[T', h']$ *be two regular systems. If* $\mathrm{Sat}(T) = \mathrm{Sat}(T')$*, then* $h'h_{T'}$ *is regular w.r.t* $\mathrm{Sat}(T)$ *and*

$$\mathbf{Z}(T, h) \setminus \mathbf{Z}(T', h') = \mathbf{Z}(h'h_{T'}, T, hh_T).$$

PROOF. Since $\mathrm{Sat}(T) = \mathrm{Sat}(T')$ and $h'h_{T'}$ is regular w.r.t $\mathrm{Sat}(T')$, $h'h_{T'}$ is regular w.r.t $\mathrm{Sat}(T)$. By Lemma 2 and Lemma 3, we have

$$\begin{aligned}
\mathbf{Z}(T, hh'h_{T'}) &= \mathbf{W}(T) \setminus \mathbf{V}(hh'h_{T'}) \\
&= \overline{\mathbf{W}(T)} \setminus \mathbf{V}(hh'h_T h_{T'}) \\
&= \overline{\mathbf{W}(T')} \setminus \mathbf{V}(hh'h_T h_{T'}) \\
&= \mathbf{W}(T') \setminus \mathbf{V}(hh'h_T) \\
&= \mathbf{Z}(T', hh'h_T).
\end{aligned}$$

Then, we can decompose $\mathbf{Z}(T, h)$ into the disjoint union

$$\mathbf{Z}(T, h) = \mathbf{Z}(T, hh'h_{T'}) \bigsqcup \mathbf{Z}(h'h_{T'}, T, hh_T).$$

Similarly, we have:

$$\mathbf{Z}(T', h') = \mathbf{Z}(T', hh'h_T) \bigsqcup \mathbf{Z}(hh_T, T', h'h_{T'}).$$

The conclusion follows from the fact that

$$\mathbf{Z}(T, hh'h_{T'}) \setminus \mathbf{Z}(T', hh'h_T) = \emptyset \quad \text{and} \quad \mathbf{Z}(h'h_{T'}, T, hh_T) \cap \mathbf{Z}(T', h') = \emptyset.$$

Lemma 6 *Assume that* $\text{Sat}(T_{<v}) = \text{Sat}(T'_{<v})$. *Then*

(i) *if* $p' := T'_v$ *is defined but not* T_v, *then* p' *is regular w.r.t* $\text{Sat}(T)$ *and*

$$\mathbf{Z}(T,h) \setminus \mathbf{Z}(T',h') = \mathbf{Z}(T,hp') \bigsqcup (\mathbf{Z}(p',T,hh_T) \setminus \mathbf{Z}(T',h')) .$$

(ii) *if* $p := T_v$ *is defined but not* T'_v, *then* p *is regular w.r.t* $\text{Sat}(T')$ *and*

$$\mathbf{Z}(T,h) \setminus \mathbf{Z}(T',h') = \mathbf{Z}(T,h) \setminus \mathbf{Z}(p,T',h'h_{T'}).$$

PROOF. (i) As $\text{init}(p')$ is regular w.r.t $\text{Sat}(T'_{<v})$, it is also regular w.r.t $\text{Sat}(T_{<v})$. Since T_v is not defined, we know $v \notin \text{mvar}(T)$. Therefore, p' is also regular w.r.t $\text{Sat}(T)$. On the other hand, we have a disjoint decomposition

$$\mathbf{Z}(T,h) = \mathbf{Z}(T,hp') \bigsqcup \mathbf{Z}(p',T,hh_T).$$

By the definition of p', $\mathbf{Z}(T',h') \subseteq \mathbf{V}(p')$ which implies

$$\mathbf{Z}(T,hp') \bigcap \mathbf{Z}(T',h') = \emptyset.$$

The conclusion follows.

(ii) Similarly, we know p is regular w.r.t $\text{Sat}(T')$. By the disjoint decomposition

$$\mathbf{Z}(T',h') = \mathbf{Z}(T',h'p) \bigsqcup \mathbf{Z}(p,T',h'h_{T'}),$$

and $\mathbf{Z}(T,h) \cap \mathbf{Z}(T',h'p) = \emptyset$, we have

$$\mathbf{Z}(T,h) \setminus \mathbf{Z}(T',h') = \mathbf{Z}(T,h) \setminus \mathbf{Z}(p,T',h'h_{T'}),$$

from which the conclusion follows.

Lemma 7 *Assume that* $\text{Sat}(T_{<v}) = \text{Sat}(T'_{<v})$ *but* $\text{Sat}(T_{\leqslant v}) \neq \text{Sat}(T'_{\leqslant v})$ *and that* v *is algebraic w.r.t both* T *and* T'. *Define*

$$\begin{aligned}
\mathcal{G} &= \mathbf{GCD}(T_v, T'_v, T_{<v}); \\
\mathcal{E} &= \bigcup_{(g,C)\in\mathcal{G},\ |C|>|T_{<v}|} \mathbf{Extend}(C, T_{\geqslant v}); \\
\mathcal{R} &= \bigcup_{E\in\mathcal{E}} \mathbf{Regularize}(hh_T, E).
\end{aligned}$$

Then we have

(i)

$$\begin{aligned}
&\mathbf{Z}(T,h) \\
&= \left(\bigcup_{R\in\mathcal{R},\ hh_T\notin\text{Sat}(R)} \mathbf{Z}(R, hh_T) \right) \bigcup \left(\bigcup_{(g,C)\in\mathcal{G},\ |C|=|T_{<v}|} \mathbf{Z}(C \cup T_{\geqslant v}, hh_T) \right).
\end{aligned}$$

(ii) rank$(R) <_r$ rank(T), *for all* $R \in \mathcal{R}$.
(iii) *Assume that* $|C| = |T_{<v}|$. *Then*
$(iii.a)$ $C \cup T_{\geqslant v}$ *is a regular chain and* hh_T *is regular w.r.t it.*
$(iii.b)$ *if* $|\mathcal{G}| > 1$, *then* rank$(C \cup T_{\geqslant v}) <_r$ rank(T).

PROOF. By the specification of $\mathbf{GCD}$ we have

$$\mathbf{W}(T_{<v}) \subseteq \bigcup_{(g,C)\in\mathcal{G}} \mathbf{W}(C) \subseteq \overline{\mathbf{W}(T_{<v})}.$$

That is,

$$\mathbf{W}(T_{<v}) \xrightarrow{D} (\mathbf{W}(C), (g, C) \in \mathcal{G}).$$

From the specification of $\mathbf{Extend}$ we have: for each $(g, C) \in \mathcal{G}$ such that $|C| > |T_{<v}|$,

$$\mathbf{W}(C \cup T_{\geqslant v}) \xrightarrow{D} (\mathbf{W}(E), E \in \mathbf{Extend}(C \cup\ T_{\geqslant v})).$$

From the specification of $\mathbf{Regularize}$, we have for all $(g, C) \in \mathcal{G}$ such that $|C| > |T_{<v}|$ and all $E \in \mathbf{Extend}(C \cup T_{\geqslant v})$,

$$\mathbf{W}(E) \xrightarrow{D} (\mathbf{W}(R),\ R \in \mathbf{Regularize}(hh_T, E)).$$

Therefore, by applying the Lifting Theorem [13] we have:

$$\begin{aligned}
\mathbf{W}(T) &= \mathbf{W}(T_{<v} \cup T_{\geqslant v}) \\
&\subseteq \left(\bigcup_{R\in\mathcal{R}} \mathbf{W}(R)\right) \bigcup \left(\bigcup_{(g,C)\in\mathcal{G},\ |C|=|T_{<v}|} \mathbf{W}(C \cup T_{\geqslant v})\right) \\
&\subseteq \overline{\mathbf{W}(T_{<v} \cup T_{\geqslant v})} \\
&= \overline{\mathbf{W}(T)},
\end{aligned}$$

which implies,

$$\begin{aligned}
&\mathbf{Z}(T, h) = \mathbf{Z}(T, hh_T) \\
&\subseteq \left(\bigcup_{R\in\mathcal{R},\ hh_T\notin \mathrm{Sat}(R)} \mathbf{Z}(R, hh_T)\right) \bigcup \left(\bigcup_{(g,C)\in\mathcal{G},\ |C|=|T_{<v}|} \mathbf{Z}(C \cup T_{\geqslant v}, hh_T)\right) \\
&\subseteq \overline{\mathbf{W}(T)} \setminus \mathbf{V}(hh_T) = \mathbf{Z}(T, h).
\end{aligned}$$

So (i) holds. If $|C| > |T_{<v}|$, by the specifications of $\mathbf{Extend}$ and $\mathbf{Regularize}$, $|R| > |T|$. By Lemma 3,

$$\dim(\mathbf{V}(\mathrm{Sat}(R))) < \dim(\mathbf{V}(\mathrm{Sat}(T))),$$

which implies (ii).

If $|C| = |T_{<v}|$, by Proposition 5 of [13], we conclude $(iii.a)$ holds. When $|\mathcal{G}| > 1$, by Notation 1, $(iii.b)$ holds.

Lemma 8 *Assume that* $\text{Sat}(T_{<v}) = \text{Sat}(T'_{<v})$ *but* $\text{Sat}(T_{\leqslant v}) \neq \text{Sat}(T'_{\leqslant v})$ *and that* v *is algebraic w.r.t both* T *and* T'*. Define* $p = T_v$, $p' = T'_v$ *and*

$$\mathcal{G} = \mathbf{GCD}(p, p', T_{<v}).$$

If $|\mathcal{G}| = 1$*, let* $\mathcal{G} = \{(g, C)\}$*. Then the following properties hold*

(i) $C = T_{<v}$.

(ii) *If* $g \in \mathbb{K}$*, then*

$$\mathbf{Z}(T, h) \setminus \mathbf{Z}(T', h') = \mathbf{Z}(T, h).$$

(iii) *If* $g \notin \mathbb{K}$ *and* $\text{mvar}(g) < v$*, then* g *is regular w.r.t* $\text{Sat}(T)$ *and*

$$\begin{aligned}&\mathbf{Z}(T, h) \setminus \mathbf{Z}(T', h')\\ &= \mathbf{Z}(T, gh) \bigsqcup (\mathbf{Z}(g, T, hh_T) \setminus \mathbf{Z}(T', h'))\,.\end{aligned}$$

(iv) *Assume that* $\text{mvar}(g) = v$.

$(iv.a)$ *If* $\text{mdeg}(g) = \text{mdeg}(p)$*, defining*

$$\begin{aligned}q' &= \text{pquo}(p', p, T'_{<v})\\ D'_p &= T'_{<v} \cup \{p\} \cup T'_{>v}\\ D'_{q'} &= T'_{<v} \cup \{q'\} \cup T'_{>v},\end{aligned}$$

then we have

$$\mathbf{Z}(T, h) \setminus \mathbf{Z}(T', h') = \mathbf{Z}(T, h) \setminus \mathbf{Z}(D'_p, h'h_{T'}),$$

$\text{rank}(D'_p) < \text{rank}(T')$ *and* $h'h_{T'}$ *is regular w.r.t* $\text{Sat}(D'_p)$.

$(iv.b)$ *If* $\text{mdeg}(g) < \text{mdeg}(p)$*, defining*

$$\begin{aligned}q &= \text{pquo}(p, g, T_{<v})\\ D_g &= T_{<v} \cup \{g\} \cup T_{>v}\\ D_q &= T_{<v} \cup \{q\} \cup T_{>v},\end{aligned}$$

then we have: D_g *and* D_q *are regular chains such that* $\text{rank}(D_g) < \text{rank}(T)$, $\text{rank}(D_q) < \text{rank}(T)$, hh_T *is regular w.r.t* $\text{Sat}(D_g)$ *and* $\text{Sat}(D_q)$*, and*

$$\mathbf{Z}(T, h) = \mathbf{Z}(D_g, hh_T) \bigcup \mathbf{Z}(D_q, hh_T) \bigcup \mathbf{Z}(h_g, T, hh_T).$$

PROOF. Since $|\mathcal{G}| = 1$, by the specification of the operation $\mathbf{GCD}$ and Notation 1, (i) holds. Therefore we have

$$\text{Sat}(C) = \text{Sat}(T_{<v}) = \text{Sat}(T'_{<v}) \tag{1}$$

There exist polynomials A and B such that

$$g \equiv Ap + Bp' \mod \text{Sat}(C). \tag{2}$$

From (2), we have

$$\mathbf{V}(\text{Sat}(C)) \subseteq \mathbf{V}(g - Ap - Bp') \tag{3}$$

Therefore, we deduce

$$\begin{aligned}
&\mathbf{W}(T) \bigcap \mathbf{W}(T') \\
&= \mathbf{W}(T_{<v} \cup p \cup T_{\geqslant v}) \bigcap \mathbf{W}(T'_{<v} \cup p' \cup T'_{\geqslant v}) \\
&\subseteq (\mathbf{W}(T_{<v}) \cap \mathbf{V}(p)) \bigcap (\mathbf{W}(T'_{<v}) \cap \mathbf{V}(p')) \\
&\subseteq \mathbf{V}(\text{Sat}(T_{<v})) \bigcap \mathbf{V}(p) \bigcap \mathbf{V}(p') && \text{by (1)} \\
&\subseteq \mathbf{V}(g - Ap - Bp') \bigcap \mathbf{V}(p) \bigcap \mathbf{V}(p') && \text{by (3)} \\
&\subseteq \mathbf{V}(g).
\end{aligned}$$

that is

$$\mathbf{W}(T) \bigcap \mathbf{W}(T') \subseteq \mathbf{V}(g). \tag{4}$$

Now we prove (ii). When $g \in \mathbb{K}$, $g \neq 0$, from (4) we deduce

$$\mathbf{W}(T) \bigcap \mathbf{W}(T') = \emptyset. \tag{5}$$

Thus we have

$$\begin{aligned}
&\mathbf{Z}(T, h) \setminus \mathbf{Z}(T', h') \\
&= (\mathbf{W}(T) \setminus \mathbf{V}(h)) \setminus (\mathbf{W}(T') \setminus \mathbf{V}(h')) \\
&= (\mathbf{W}(T) \setminus \mathbf{V}(h)) && \text{by (5)} \\
&= \mathbf{Z}(T, h).
\end{aligned}$$

Now we prove (iii). Since $C = T_{<v}$ and mvar(g) is smaller than or equal to v, by the specification of $\mathbf{GCD}$, g is regular w.r.t Sat(T). We have following decompositions

$$\begin{aligned}
\mathbf{Z}(T, h) &= \mathbf{Z}(T, gh) \bigsqcup \mathbf{Z}(g, T, hh_T), \\
\mathbf{Z}(T', h') &= \mathbf{Z}(T', gh') \bigsqcup \mathbf{Z}(g, T', h'h_{T'}).
\end{aligned}$$

On the other hand,

$$\begin{aligned}
&\mathbf{Z}(T, gh) \bigcap \mathbf{Z}(T', gh') \\
&= (\mathbf{W}(T) \cap \mathbf{V}(gh)^c) \bigcap (\mathbf{W}(T') \cap \mathbf{V}(gh')^c) \\
&\subseteq (\mathbf{W}(T) \cap \mathbf{V}(g)^c) \bigcap (\mathbf{W}(T') \cap \mathbf{V}(g)^c) \\
&= (\mathbf{W}(T) \cap \mathbf{W}(T')) \bigcap \mathbf{V}(g)^c \\
&= \emptyset \qquad \text{by } (4).
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\mathbf{Z}(T, h) \setminus \mathbf{Z}(T', h') \\
&= (\mathbf{Z}(T, gh) \setminus \mathbf{Z}(T', gh')) \bigsqcup (\mathbf{Z}(g, T, hh_T) \setminus \mathbf{Z}(T', h')) \\
&= \mathbf{Z}(T, gh) \bigsqcup (\mathbf{Z}(g, T, hh_T) \setminus \mathbf{Z}(T', h')) .
\end{aligned}$$

Now we prove $(iv.a)$. First, both h and h'_T are regular w.r.t Sat(C) = Sat$(T_{<v})$ = Sat$(T'_{<v})$. From the construction of D'_p, we have $hh_{T'}$ is regular w.r.t Sat(D'_p).

Assume that mvar$(g) = v$ and mdeg(g) = mdeg(p). We note that mdeg$(p') >$ mdeg(p) holds. Otherwise we would have mdeg(g) = mdeg(p) = mdeg(p') which implies:

$$p \in \text{Sat}(T'_{\geqslant v}) \text{ and } p' \in \text{Sat}(T_{\geqslant v}). \tag{6}$$

Thus

$$\begin{aligned}
\text{Sat}(T_{\leqslant v}) &= \langle T_{\leqslant v} \rangle : h^{\infty}_{T_{\leqslant v}} = \langle T_{<v} \cup p \rangle : h^{\infty}_{T_{\leqslant v}} \\
&\subseteq \text{Sat}(T'_{\leqslant v}) : h^{\infty}_{T_{\leqslant v}} \qquad \text{by (6)} \\
&= \text{Sat}(T'_{\leqslant v}),
\end{aligned}$$

that is Sat$(T_{\leqslant v}) \subseteq$ Sat$(T'_{\leqslant v})$. Similarly, Sat$(T'_{\leqslant v}) \subseteq$ Sat$(T_{\leqslant v})$ holds. So we have Sat$(T'_{\leqslant v})$ = Sat$(T_{\leqslant v})$, a contradiction.

Hence, mvar$(q') = v$.

By Lemma 6 [13], we know that D'_p and $D'_{q'}$ are regular chains. Then with Theorem 7 [13] and Lifting Theorem [13], we know

$$\begin{aligned}
\mathbf{Z}(T', h') &\subseteq \mathbf{Z}(D'_p, h') \bigcup \mathbf{Z}(D'_{q'}, h') \bigcup \mathbf{Z}(h_p, T', h') \\
&\subseteq \overline{\mathbf{W}(T')} \setminus \mathbf{V}(h').
\end{aligned}$$

By Lemma 2, we have

$$\mathbf{Z}(T', h') = \mathbf{Z}(D'_p, h'h_{T'}) \bigcup \mathbf{Z}(D'_{q'}, h'h_{T'}) \bigcup \mathbf{Z}(h_p, T', h'h_{T'}).$$

Since

$$\begin{aligned}
\mathbf{Z}(D'_{q'}, h'h_{T'}) &= \mathbf{Z}(D'_{q'}, h_p h' h_{T'}) \bigcup \mathbf{Z}(h_p, D'_{q'}, h'h'_T) \\
&= \mathbf{Z}(D'_{q'}, p h_p h' h_{T'}) \bigcup \mathbf{Z}(p, D'_{q'}, h_p h' h'_T) \bigcup \mathbf{Z}(h_p, D'_{q'}, h'h'_T)
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{Z}(p, D'_{q'}, h_p h' h'_T) &\subseteq \mathbf{Z}(D'_p, h'h_{T'}) \\
\mathbf{Z}(h_p, D'_{q'}, h'h'_T) &\subseteq \mathbf{Z}(h_p, T', h'h_{T'}),
\end{aligned}$$

we deduce

$$\mathbf{Z}(T', h') = \mathbf{Z}(D'_p, h'h_{T'}) \bigsqcup \mathbf{Z}(D'_{q'}, p h' h_{T'}) \bigsqcup \mathbf{Z}(h_p, T', h'h_{T'}).$$

Now observe that

$$\begin{aligned}
&\mathbf{Z}(T, h) \bigcap \mathbf{Z}(D'_{q'}, p h' h_{T'}) = \emptyset, \text{ and} \\
&\mathbf{Z}(T, h) \bigcap \mathbf{Z}(h_p, T', h'h_{T'}) = \emptyset.
\end{aligned}$$

We obtain

$$\mathbf{Z}(T,h) \setminus \mathbf{Z}(T',h') = \mathbf{Z}(T,h) \setminus \mathbf{Z}(D'_p, h'h_{T'}).$$

Finally we prove $(iv.b)$. We assume that mvar$(g) = v$ and mdeg$(g) <$ mdeg(p); this implies mvar$(q) = v$. Applying Lemma 6 in [13] we know that D_g and D_q are regular chains and satisfy the desired rank condition. Then by Theorem 7 [13] and Lifting Theorem [13] we have

$$\mathbf{Z}(T,h) = \mathbf{Z}(D_g, hh_T) \bigcup \mathbf{Z}(D_q, hh_T) \bigcup \mathbf{Z}(h_g, T, hh_T).$$

This completes the whole proof.

Definition 5 *Given two pairs of ranks* $(\text{rank}(T_1), \text{rank}(T'_1))$ *and* $(\text{rank}(T_2), \text{rank}(T'_2))$*, where* T_1, T_2, T'_1, T'_2 *are triangular sets. We define the product order* $<_p$ *of Ritt order* $<_r$ *on them as follows*

$$\begin{aligned}&(\text{rank}(T_2), \text{rank}(T'_2)) <_p (\text{rank}(T_1), \text{rank}(T'_1))\\ \iff &\begin{cases} \text{rank}(T_2) <_r \text{rank}(T_1) \ \textit{or} \\ \text{rank}(T_2) = \text{rank}(T_1),\ \text{rank}(T'_2) <_r \text{rank}(T'_1). \end{cases}\end{aligned}$$

In the following theorems, we prove the termination and correctness separately. Along with the proof of Theorem 1, we show the rank conditions are satisfied which is part of the correctness. The remained part, say zero set decomposition, will be proved in Theorem 2.

Theorem 1 *Algorithms* **Difference** *and* **DifferenceLR** *terminate and satisfy the rank conditions in their specifications.*

PROOF. It is equivalent to prove that

(i) **Difference** terminates with rank(**Difference**$([T,h],[T',h'])) \leqslant_r$ rank$([T,h])$,
(ii) **DifferenceLR** terminates with rank(**DifferenceLR**$(\mathcal{L},\mathcal{R})) \leqslant_r$ rank$(\mathcal{L})$.

(1) Basic case: no recursive calls to **Difference** and **DifferenceLR**.
First, (ii) holds for the algorithm **DifferenceLR**:
- Line 2 by $\mathcal{L} = \emptyset$, rank$(\emptyset) <_r$ rank$(\mathcal{L})$,
- Line 4 by $\mathcal{R} = \emptyset$, rank$(\mathcal{L}) =$ rank$(\mathcal{L})$,
- Line $2, 4$ the termination is obvious.

Next, (i) holds for the algorithm **Difference**:
- Line 2 by Lemma 4, **Intersect** terminates and

$$\text{rank}(\mathbf{Intersect}(h'h_{T'}, T, hh_T)) <_r \text{rank}([T,h]),$$

- Line $7, 8$ when $\mathbf{Intersect}(p', T, hh_T) = \emptyset$, we conclude (i) holds from (ii) and rank$([T, hp'])$=rank(T),
- Line 11 when $\mathbf{Intersect}(p, T', h'h_{T'}) = \emptyset$, we conclude (i) holds from (ii) and rank$([T,h])$=rank$([T,h])$,

- Line 18 rank($[T, h]$) = rank($[T, h]$),
- Line 20, 21 when $\mathbf{Intersect}(g, T, hh_T) = \emptyset$, we conclude (i) holds from (ii) and rank($[T, gh]$)=rank($[T, h]$).

(2) Induction hypothesis: assume that both (i) and (ii) hold for inputs with ranks smaller than the rank of $([T, h], [T', h'])$ w.r.t. $<_p$.

(3) By (1), if no recursive calls occur in one branch, then (i) and (ii) already hold. When recursive calls occur, we first prove (i) holds. Indeed by induction (2), it suffices to prove: the inputs of recursive calls to **Difference** or **DifferenceLR** have smaller ranks than rank($([T, h], [T', h'])$) w.r.t $<_p$. We show this line by line for **Difference**:

- Line 8 by Lemma 6 and Lemma 4,

$$\text{rank}(\mathbf{Intersect}(p', T, hh_T)) <_r \text{rank}([T, h]),$$

- Line 11 by Lemma 6 and Lemma 4,

$$\text{rank}(\mathbf{Intersect}(p, T', h'h_{T'})) <_r \text{rank}([T', h']),$$

- Line 21 by Lemma 8 and Lemma 4,

$$\text{rank}(\mathbf{Intersect}(g, T, hh_T)) <_r \text{rank}([T, h]),$$

- Line 25 by Lemma 8,

$$\text{rank}([D'_p, h'h_{T'}]) <_r \text{rank}([T', h']),$$

- Line 30, 31 by Lemma 8,

$$\text{rank}([D_g, hh_T]) <_r \text{rank}([T, h]) \text{ and } \text{rank}([D_q, hh_T]) <_r \text{rank}([T, h]),$$

- Line 32 by Lemma 8 and Lemma 4,

$$\text{rank}(\mathbf{Intersect}(h_g, T, hh_T)) <_r \text{rank}([T, h]),$$

- Line 41, 46 by Lemma 7,

$$\text{rank}([D, hh_T]) <_r \text{rank}([T, h]) \text{ and } \text{rank}([C \cup T_{\geqslant v}, hh_T]) <_r \text{rank}([T, h]).$$

Now we prove, when recursive calls occur, (ii) holds for **DifferenceLR** as well. Since (i) holds for **Difference**, from the algorithm **DifferenceLR**, clearly it terminates. So by (2), it suffices to prove: the rank of the input of each recursive call to **Difference** is less than or equal to rank((L, R)). We also show this line by line for **DifferenceLR**:

- Line 8 by rank($[T, h]$) $<_r$ rank(L),
- Line 15 by (i),

$$\text{rank}(\mathbf{Difference}([T, h], [T', h'])) \leqslant_r \text{rank}([T, h]) \leqslant_r \text{rank}(\mathcal{L}).$$

Theorem 2 *Both* **Difference** *and* **Difference** *satisfy their specifications.*

PROOF. By Theorem 1, **Difference** and **Difference** terminate and satisfy their rank conditions. So it suffices to prove the correctness of **Difference** and **DifferenceLR**, that is

(i) $\mathbf{Z}(T,h) \setminus \mathbf{Z}(T',h') = \mathbf{Z}(\mathbf{Difference}([T,h],[T',h']))$,
(ii) $\mathbf{Z}(\mathcal{L}) \setminus \mathbf{Z}(\mathcal{R}) = \mathbf{Z}(\mathbf{DifferenceLR}(\mathcal{L},\mathcal{R}))$.

Similarly we prove the correctness of the two algorithms by induction.

(1) Basic case: no recursive call to **Difference** and **DifferenceLR** occurs.
First, (ii) holds for algorithm **DifferenceLR**.
- Line 2 $\mathbf{Z}(\emptyset) \setminus \mathbf{Z}(\mathcal{R}) = \mathbf{Z}(\emptyset)$,
- Line 4 $\mathbf{Z}(\mathcal{L}) \setminus \emptyset = \mathbf{Z}(\mathcal{L})$.

Next, (i) holds for algorithm **Difference**.
- Line 2 by Lemma 4 and Lemma 5,
- Line $7, 8$ by Lemma 4, Lemma 6 and (ii),
- Line 11 by Lemma 4, Lemma 6 and (ii),
- Line 18 by Lemma 8,
- Line $20, 21$ by Lemma 4, Lemma 6 and (ii).

(2) Induction hypothesis: both (i) and (ii) hold for inputs with ranks smaller than rank$([T,h],[T',h'])$ w.r.t. $<_p$.

(3) By (1), if no recursive call occurs, (i) and (ii) already hold. When recursive call occurs, we first show (i) holds. From the proof of Theorem 1, in **Difference**, the inputs of recursive calls to **Difference** and **DifferenceLR** will have smaller ranks w.r.t. the product order $<_p$. Therefore, by (2) we show prove (i) holds for **Difference** line by line
- Line $7, 8$ by Lemma 4 and Lemma 6,
- Line 11 by Lemma 4 and Lemma 6,
- Line $20, 21$ by Lemma 4 and Lemma 8,
- Line 25 by Lemma 8,
- Line $30, 31, 32$ by Lemma 4 and Lemma 8,
- Line 32 by Lemma 8 and Lemma 4,

$$\text{rank}(\mathbf{Intersect}(h_g, T, hh_T)) <_r \text{rank}([T,h]),$$

- Line $41, 46$ by Lemma 7.

Finally, when recursive call occurs, (ii) holds for **DifferenceLR** line by line.
- Line $5-9$ by (i) and the relation

$$\begin{aligned}
&\mathbf{Z}(\mathcal{L}) \setminus \mathbf{Z}(\mathcal{R}) \\
&= \left(\bigcup_{i=1}^{r} \mathbf{Z}(T_i,h_i)\right) \setminus \mathbf{Z}(T',h') = \bigcup_{i=1}^{r} \left(\mathbf{Z}(T_i,h_i) \setminus \mathbf{Z}(T',h')\right) \\
&= \bigcup_{i=1}^{r} \mathbf{Z}\left(\mathbf{Difference}([T_i,h_i],[T',h'])\right)
\end{aligned}$$

– Line $10-18$ by (i),

$$\mathbf{Z}(\mathbf{Difference}([T,h],[T',h'])) = \mathbf{Z}(T,h) \setminus \mathbf{Z}(T',h')$$

and the relation

$$\mathbf{Z}(\mathcal{L}) \setminus \mathbf{Z}(\mathcal{R}) = (\mathbf{Z}(\mathcal{L}) \setminus \mathbf{Z}(R_1,g_1)) \setminus \left(\bigcup_{j=2}^{s} \mathbf{Z}(R_j,g_j) \right),$$

this reduces to the case $|\mathcal{R}| = 1$ (Line 5–9).

6 Verification of triangular decompositions

In this section, we describe how to verify the output from a triangular decomposition. Verification in Kalkbrener's sense is still unknown whether we can circumvent Gröbner basis computations. However, in Lazard's sense, we will present both Gröbner basis and triangular decomposition methods.

6.1 Verification with Gröbner bases

The following two lemmas state the Gröbner basis methods to verify whether two basic constructible sets are equal or not.

Lemma 1. *Let $\{F,f\}$ and $\{G_0,g_0\}$ be two polynomial systems. The following statements are equivalent*

1. $\mathbf{D}(F,f) \setminus \mathbf{D}(G_0,g_0) \subseteq \bigcup_{i=1}^{r} \mathbf{D}(G_i,g_i)$.
2. *For every $\{i_1,\ldots,i_s\} \subseteq \{0,\ldots,r\}$, $0 \le s \le r$,*

$$\sqrt{\langle F \cup \{g_{i_1},\ldots,g_{i_s}\}\rangle} \supseteq \prod_{k\in\{0,\ldots,r\}\setminus\{i_1,\ldots,i_s\}} \langle f\rangle\langle G_k\rangle. \tag{7}$$

PROOF. (1) is equivalent to $\mathbf{D}(F,f) \subseteq \bigcup_{i=0}^{r} \mathbf{D}(G_i,g_i)$.

$$\mathbf{D}(F,f) \bigcap \left(\bigcap_{i=0}^{e} \mathbf{D}(G_i,g_i)^c \right) = \emptyset.$$

Using the distributive property, we deduce that (1) is equivalent to

$$\left(\mathbf{D}(F,f) \bigcap \mathbf{V}(g_{i_1},\ldots,g_{i_s})\right) \bigcap \left(\bigcap_{k\in\{0,\ldots,r\}\setminus\{i_1,\ldots,i_s\}} \mathbf{V}(G_k)^c \right) = \emptyset,$$

for all subsets $\{i_1,\ldots,i_s\}$ of $\{0,\ldots,r\}$. The proof easily follows.

Lemma 2. *Let $\{F,f\}$ and $\{G,g\}$ be two polynomial systems. The following statements are equivalent*

1. $\mathbf{D}(F, f) \setminus \mathbf{D}(G, g) \supseteq \bigcup_{i=1}^{r} \mathbf{D}(H_i, h_i)$.
2. *For all* $1 \leq i \leq r$*, we have*

$$h_i g \in \sqrt{\langle H_i \cup G \rangle}, h_i \in \sqrt{\langle H_i, f \rangle}, \text{ and } \langle h_i \rangle \langle F \rangle \subset \sqrt{\langle H_i \rangle}. \tag{8}$$

PROOF. (1) holds if and only if for each $1 \leq i \leq r$ we have

$$\begin{cases} \mathbf{D}(H_i, h_i) \bigcap \mathbf{D}(F, f)^c = \emptyset, \\ \mathbf{D}(H_i, h_i) \bigcap \mathbf{D}(G, g) = \emptyset, \end{cases}$$

which holds if and only if

$$\begin{cases} \mathbf{V}(H_i) \bigcap \mathbf{V}(h_i)^c \bigcap \mathbf{V}(F)^c = \emptyset, \\ \mathbf{V}(H_i) \bigcap \mathbf{V}(h_i)^c \bigcap \mathbf{V}(f) = \emptyset, \\ \mathbf{V}(H_i) \bigcap \mathbf{V}(h_i)^c \bigcap \mathbf{V}(G) \bigcap \mathbf{V}(g)^c = \emptyset. \end{cases}$$

The proof easily follows.

6.2 Verification with triangular decompositions

Given two Lazard's triangular decompositions $\{T_i \mid i = 1 \ldots e\}$ and $\{S_j \mid j = 1 \ldots f\}$. Checking $\cup_{i=1}^{e} \mathbf{W}(T_i) = \cup_{j=1}^{f} \mathbf{W}(S_j)$ amounts to checking both

$$\left(\bigcup_{i=1}^{e} \mathbf{W}(T_i) \right) \setminus \left(\bigcup_{j=1}^{f} \mathbf{W}(S_j) \right) \text{ and } \left(\bigcup_{j=1}^{f} \mathbf{W}(S_j) \right) \setminus \left(\bigcup_{i=1}^{e} \mathbf{W}(T_i) \right)$$

being empty. In turn, after computing the regular system representations of above two constructible sets. According to Lemma 1, we solve the verification problem with the algorithm $\mathbf{DifferenceLR}$ in Lazard's sense.

7 Experimentation

We have implemented a verifier, named *Diff-verifier*, according to the $\mathbf{DifferenceLR}$ algorithm proposed in Section 5, and it has been implemented in Maple 11 based on the `RegularChains` library. To verify the effectiveness of our Diff-verifier, we have also implemented another verifier, named *GB-verifier*, applying Lemma 9 and 10, on top of the *PolynomialIdeals* package in Maple 11.

We use these two verifiers to examine four polynomial system solvers herein. They are the *Triangularize* function in the *RegularChains* library [11], the TRIADE server in *Aldor*, written with the *BasicMath library* [8], the *RegSer* function and the *SimSer* function in *Epsilon* [19] implemented in *Maple*. The first two solvers solve a polynomial system into regular chains by means of the TRIADE algorithm [13]. They can work in both *Lazard's* sense and *Kalkbrener's* sense. In this work, we use the options for solving in *Lazard's* sense. The *RegSer* function decomposes a polynomial system into regular systems in the sense of [22], and the *SimSer* function decomposes a polynomial system into simple systems, as in [20].

The problems used in this benchmark are chosen from [12, 17, 19]. In Table 1, for each system, we give the $dimension$ sequence of the triangular decomposition computed in *Kalkbrener's* sense by the TRIADE algorithm. The number of variables is denoted by n, and d is the maximum degree of a monomial in the input. We also give the number of components in the solution set for each of the methods we are studying.

Table 2 gives the timing of each problem solved by the four methods. In this study, due to the current availability of *Epsilon*, the timings obtained by the *RegSer* and the *SimSer* commands are performed in Maple 8 on Intel Pentium 4 machines (1.60GHz CPU, 513MB memory and Red Hat Linux 3.2.2-5). All the other timings are run on Intel Pentium 4 (3.20GHz CPU, 2.0GB total memory, and Red Hat 4.0.0-9), and the Maple version used is 11. The TRIADE server is a stand-alone executable program compiled from a program in Aldor.

Table 3 summarizes the timings of GB-verifier for verifying the solutions of the four methods. Table 4 illustrates the timings of Diff-verifier for checking the solutions by Maple *Triangularize* against Aldor TRIADE server, Maple *Triangularize* against Epsilon *RegSer*, and Epsilon *RegSer* against Epsilon *SimSer*. For the case where there is a time, the verifying result is also true. The $'-'$ denotes the case where the test stalls by either reaching the time limit of 43200 seconds or causing a memory failure.

This experimentation results illustrate that verifying a polynomial solver is a truly difficult task. The GB-verifier is very costly in terms of cpu time and memory. It only succeeds for some easy examples. Assuming that the GB-verifier is reliable, for the examples it succeeds, the Diff-verifier agrees with its results by pair-wise checking, while it takes much less time. This shows the efficiency of our Diff-verifier. Further more, the tests also show that the Diff-verifier can verify more difficult problems by pair-wise checking. The tests indicate that all of the four methods are solving tools with a high probability of correctness, since the checking results would not agree to each other otherwise.

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. P. Aubry and M. Moreno Maza. Triangular sets for solving polynomial systems: A comparative implementation of four methods. *J. Symb. Comp.*, 28(1-2):125–154, 1999.
3. J. Backelin and R. Fröberg. How we proved that there are exactly 924 cyclic 7-roots. In S. M. Watt, editor, *Proc. ISSAC'91*, pages 103–111. ACM, 1991.
4. F. Boulier, F. Lemaire, and M. Moreno Maza. Well known theorems on triangular systems and the D5 principle. In *Proc. of* Transgressive Computing 2006, Granada, Spain, 2006.
5. L. Donati and C. Traverso. Experimenting the Gröbner basis algorithm with the ALPI system. In *Proc. ISSAC'89*, pages 192–198. ACN Press, 1989.
6. D. Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*. Springer-Verlag, New York-Berlin-Heidelberg, 1995.
7. J. Grabmeier, E. Kaltofen, and V. Weispfenning, editors. *Computer Algebra Handbook*. Springer, 2003.
8. The Computational Mathematics Group. The basicmath library. NAG Ltd, Oxford, UK, 1998. http://www.nag.co.uk/projects/FRISCO.html.
9. R. Hartshorne. *Algebraic Geometry*. Springer-Verlag, 1997.

					Number of Components			
Sys	Name	n	d	Dimension	Maple Triangularize	Aldor TRIADE server	Epsilon RegSer	Epsilon SimSer
1	Montes S1	4	2	[2,2,1]	3	3	3	3
2	Montes S2	4	3	[0]	1	1	1	1
3	Montes S3	3	3	[1,1]	2	2	2	3
4	Montes S4	4	2	[0]	1	1	1	1
5	Montes S6	4	3	[2,2,2]	3	3	3	3
6	Montes S7	4	3	[1]	2	2	3	6
7	Montes S8	4	12	[2,1]	2	2	6	6
8	Alonso	7	4	[3]	3	3	3	4
9	Raksanyi	8	3	[4]	4	4	4	10
10	YangBaxter Rosso	6	3	[4,3,3,1,1,1,1]	7	7	4	13
11	l-3	4	3	[0,0,0,0,0,0, 0,0,0,0,0,0,0]	25	13	8	8
12	Caprasse	4	4	[0,0,0,0,0]	15	5	4	4
13	Reif	16	3	[ ]	0	0	0	0
14	Buchberger WuWang	5	3	[2]	3	3	3	4
15	DonatiTraverso	4	31	[1]	6	3	3	3
16	Wu-Wang.2	13	3	[1,1,1,1,1]	5	5	5	5
17	Hairer-2-BGK	13	4	[2]	4	4	5	6
18	Montes S5	8	3	[4]	4	4	4	10
19	Bronstein	4	3	[1]	4	2	4	9
20	Butcher	8	4	[3,3,3,2,2,0]	7	6	6	6
21	genLinSyst-2-2	8	2	[6]	11	11	11	11
22	genLinSyst-3-2	11	2	[8]	17	18	18	18
23	Gerdt	7	4	[3,2,2,2,1,1]	7	6	10	10
24	Wang93	5	3	[1]	5	4	6	7
25	Vermeer	5	5	[1]	5	4	12	14
26	Gonnet	5	2	[3,3,3]	3	3	9	9
27	Neural	4	3	[1,1]	4	3	–	–
28	Noonburg	4	3	[1,1]	4	3	–	–
29	KdV	1	0	[12,12,11, 11,11,11,11]	7	7	–	–
30	Montes S12	8	2	[4]	22	17	23	–
31	Pappus	12	2	[6,6,6,6,6, 6,6,6,6,6]	124	129	156	–

Table 1 Features of the polynomial systems

Sys	Maple Triangularize	Aldor TRIADE server	Epsilon RegSer	Epsilon SimSer
1	0.104	0.164	0.01	0.03
2	0.039	0.204	0.03	0.02
3	0.069	0.06	0.019	0.111
4	0.510	0.072	0.049	0.03
5	0.052	0.096	0.03	0.03
6	0.150	0.06	0.09	5.14
7	0.376	0.072	0.2	1.229
8	0.204	0.065	0.109	0.16
9	0.460	0.066	0.141	0.481
10	1.252	0.108	0.069	0.21
11	5.965	0.587	1.53	2.91
12	2.426	0.167	1.209	2.32
13	123.823	1.886	1.979	2.36
14	0.2	0.101	0.049	0.109
15	2.641	0.08	0.439	0.7
16	105.835	1.429	5.49	6.14
17	23.453	0.688	1.76	1.679
18	0.484	0.078	0.13	0.471
19	0.482	0.071	0.24	1.000
20	9.325	0.442	1.689	2.091
21	0.557	0.096	0.13	0.21
22	1.985	0.173	0.431	0.411
23	4.733	0.499	3.5	4.1
24	7.814	5.353	2.18	30.24
25	26.533	0.580	4.339	60.65
26	3.983	0.354	2.18	2.48
27	15.879	1.567	–	–
28	15.696	1.642	–	–
29	9245.442	49.573	–	–
30	17.001	0.526	2.829	–
31	79.663	4.429	11.78	–

Table 2 Solving timings in sec. of the four methods

	GB-verifier timing(s)				Diff-verifier timing(s)		
sys	Maple Triangularize (M.T.)	Aldor TRIADE server (A.T.)	Epsilon RegSer (E.R.)	Epsilon SimSer (E.S.)	M.T. vs A.T.	M.T. vs E.R.	E.R. vs E.S.
1	0.556	0.526	0.518	0.543	0.58	0.439	0.445
2	0.128	0.127	0.129	0.131	0.039	0.02	0.013
3	0.584	0.575	0.585	2.874	0.182	0.108	0.427
4	0.104	0.133	0.139	0.137	0.037	0.027	0.023
5	1.484	1.472	1.457	1.469	0.591	0.339	0.356
6	76.596	72.374	71.853	–	7.204	5.268	15.334
7	0.616	0.601	4.501	4.536	0.573	0.758	1.017
8	–	–	–	–	1.196	1.564	2.618
9	–	–	–	–	5.442	9.837	18.252
10	–	–	–	–	10.888	22.638	22.649
11	–	–	–	–	14.652	4.541	3.585
12	–	58.332	33.469	35.213	2.52	2.398	3.113
13	–	–	–	–	0	0	0
14	1.96	1.937	2.165	5.739	0.924	0.915	1.155
15	330.317	–	–	–	2.244	4.782	4.201
16	10466.587	–	–	–	4.34	4.408	3.207
17	–	–	–	–	6.348	6.109	15.719
18	–	–	–	–	5.32	10.485	17.897
19	1.544	0.717	5.046	–	7.838	7.986	43.506
20	–	–	–	–	13.04	10.218	9.978
21	–	–	–	–	10.872	15.098	11.048
22	–	–	–	–	61.147	48.865	32.184
23	–	–	–	–	11.144	15.981	16.222
24	–	–	–	–	1564.654	1918.968	870.962
25	–	–	–	–	2144.726	–	2182.401
26	–	–	–	–	3.839	6.041	9.550
27	11383.335	–	–	–	1088.563	–	–
28	–	–	–	–	1119.449	–	–
29	–	–	–	–	30.016	–	–
30	–	–	–	–	–	–	–
31	–	–	–	–	–	–	–

Table 3 Timings of GB-verifier and Diff-verifier

10. M. Kalkbrener. A generalized euclidean algorithm for computing triangular representations of algebraic varieties. *J. Symb. Comp.*, 15:143–167, 1993.
11. F. Lemaire, M. Moreno Maza, and Y. Xie. The `RegularChains` library. In Ilias S. Kotsireas, editor, Maple Conference 2005, pages 355–368, 2005.
12. Montserrat Manubens and Antonio Montes. Improving dispgb algorithm using the discriminant ideal, 2006.
13. 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.
14. J. O'Halloran and M. Schilmoeller. Gröbner bases for constructible sets. *Journal of Communications in Algebra*, 30(11), 2002.
15. P. Samuel and O. Zariski. *Commutative algebra*. D. Van Nostrand Company, INC., 1967.
16. W. Sit. Computations on quasi-algebraic sets. In R. Liska, editor, *Electronic Proceedings of IMACS ACA'98*, 1998.
17. *The SymbolicData Project*. http://www.SymbolicData.org, 2000–2006.
18. D. Wang. Computing triangular systems and regular systems. *Journal of Symbolic Computation*, 30(2):221–236, 2000.
19. D. M. Wang. *Epsilon 0.618*. http://www-calfor.lip6.fr/∼wang/epsilon.
20. D. M. Wang. Decomposing polynomial systems into simple systems. *J. Symb. Comp.*, 25(3):295–314, 1998.
21. D. M. Wang. *Elimination Methods*. Springer, Wein, New York, 2000.
22. D.M. Wang. Computing triangular systems and regular systems. *J. Symb. Comp.*, 30(2):221–236, 2000.