Data Structures and Polynomial Equation Solving*

D. Castro[1], M. Giusti[2], J. Heintz[1,3,4], G. Matera[5,6] and L.M. Pardo[1,2]

8th May 2001

Abstract

Elimination theory is at the origin of algebraic geometry in the 19-th century and deals with algorithmic solving of multivariate polynomial equation systems over the complex numbers, or, more generally, over an arbitrary algebraically closed field. In this paper we investigate the *intrinsic* sequential time complexity of *universal* elimination procedures for *arbitrary continuous* data structures encoding input and output objects of elimination theory (i.e. polynomial equation systems) and admitting the representation of certain limit objects.

Our main result is the following: let be given such a data structure and together with this data structure a universal elimination algorithm, say P, solving arbitrary parametric polynomial equation systems. Suppose that the algorithm P avoids "unnecessary" branchings and that P admits the efficient computation of certain natural limit objects (as e.g. the Zariski closure of a given constructible algebraic set or the parametric greatest common divisor of two given algebraic families of univariate polynomials). Then P cannot be a polynomial time algorithm.

The paper contains different variants of this result which are formulated and discussed both from the point of view of exact (i.e. symbolic) as well as from the point of view of approximative (i.e. numeric) computing.

*Research was partially supported by the following Argentinian, European, French and Spanish grants : UBACyT UBA X–198, UNLP X–272, PIP CONICET 4571, BMBF–SETCIP AL/A97-EXI/09, ECOS A99E06, CNRS 1026 MEDICIS, DGCYT BFM2000–0349, Cooperación Científica con Iberoamerica and HF-1999-055

[1] *Depto. de Matemáticas, Estadística y Computación, Facultad de Ciencias, Universidad de Cantabria, E-39071 Santander, Spain.*

[2] *Laboratoire GAGE, Ecole Polytechnique, F-91128 Palaiseau Cedex, France.*

[3] *Depto. de Matemáticas, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Ciudad Universitaria, Pabellón I (1428) Buenos Aires, Argentina.*

[4] *Member of the National Council of Science and Technology (CONICET), Argentina.*

[5] *Instituto de Desarrollo Humano, Universidad Nacional de General Sarmiento, Roca 850 (1663), San Miguel, Argentina.*

[6] *Departamento de Computación, Universidad Favaloro, Belgrano 1723 (1093) Buenos Aires, Argentina.*

The mentioned results shall only be discussed informally. Proofs will appear elsewhere.

Keywords. Polynomial equation solving, elimination theory, complexity, continuous data structure.

1 Introduction. Basic Notions.

Complexity theory deals with the efficiency of answering mathematical questions about mathematical objects. In this context, mathematical objects happen usually to posses a *unique* encoding in a previously fixed data structure (e.g. integers are encoded by their bit representation, polynomials by their coefficients, etc.). Once a data structure is fixed, standard complexity theory searches for an efficient algorithm answering the mathematical questions under consideration and tries to certify the optimality of this algorithm.

However, things become somewhat more complicated in the particular case of geometric elimination theory (polynomial equation solving in algebraically or real closed fields). Complexity theory for geometric elimination requires *simultaneous* optimization of data structures *and* algorithms. In order to illustrate this statement, let us consider the following first order formula, say Φ, belonging to the language of the elementary theory of algebraically closed fields of characteristic zero:

$$(\exists X_1)\cdots(\exists X_n)\big(X_1-T-1=0\wedge X_2^2-X_1=0\wedge\cdots\wedge X_n-X_{n-1}^2=0\wedge Y=X_n^2\big).$$

The formula Φ contains two free variables, namely T and Y. Moreover Φ is logically equivalent to the following quantifier–free formula, which we denote by Ψ:

$$Y-\sum_{i=0}^{2^n}\binom{2^n}{i}T^i=0.$$

If we choose as our data structure the standard dense or sparse encoding of polynomials by their coefficients, then Φ has length $O(n)$, whereas the length of Ψ exceeds 2^n. However, if we encode polynomials by *arithmetic circuits* (or straight–line programs), then Φ and Ψ happen both to be of length $O(n)$, since the polynomial $\sum_{i=0}^{2^n}\binom{2^n}{i}T^i=(1+T)^{2^n}$ can be evaluated in $n+1$ steps, using iterated squaring.

For the dense (or sparse) representation of polynomials *superexponential* (sequential) time is necessary (and sufficient) in order to eliminate a *single* quantifier block (see e.g. [CGH89], [DFGS91], [GV88], [HRS89], [Can88]), whereas the elimination of an *arbitrary* number of quantifier blocks requires *doubly exponential* time in this data structure (see [Hei83], [Wei88], [DH88], [FGM90a], [FGM90b], [MP93] for lower and upper complexity bounds and [Ier89], [HRS90], [Ren92] for upper complexity bounds only).

The existing superexponential (mostly Gröbner basis) algorithms for the elimination of a single quantifier block are often asymptotically optimal for the dense and sparse encoding of polynomials. Nevertheless their complexity makes them infeasible for real world sized problems (however not so for impressive software demos).

This situation suggests that these elimination algorithms require alternative data structures if one whishes to improve their complexity behaviour substantially. This observation led in the past to the idea to use arithmetic circuits for the representation of the polynomials occurring in the basic elimination procedures of algebraic and semialgebraic geometry (see [HS82], [HS81] and [Kal88] for an early application of this idea). This change of data structure allowed in a first attempt to reduce the complexity of the elimination of one block of quantifiers from superexponential to single exponential time ([GH93], [GHS93], [FGS95], [KP94], [KP96], [Mat99]). However, the corresponding algorithms required the dense representation of the input polynomials and returned a circuit encoding of the output polynomials. Therefore these algorithms were unadapted to successive elimination of several quantifier blocks (see [PS98] for more details) and unable to profit from a possible special geometric feature of the input system.

In a second attempt ([GHMP95], [GHM$^+$98], [GHH$^+$97], [GHMP97]), this problem could be settled by means of a new elimination procedure which transforms a given circuit representation of the input polynomial into a circuit representation of the output polynomials. The time complexity of this new procedure is roughly the circuit size of the input polynomials multiplied by a *polynomial* function of a certain geometric invariant of the input system, called its *degree*. Let us observe that the degree is always bounded by the Bézout–number of the input system and happens often to be considerably smaller.

For worst case input systems, the new algorithm becomes polynomial in the Bézout–number of the system, and this was the first time that this complexity goal could be reached without hiding an exponential extra factor (compare [MP97], [Roj00]).

Afterwards the new algorithm and its data structure was extended and refined in [HKP$^+$00], [GS99], [HMPS00], [HMW01], [GLS01], [Sch00], [Lec01], and in [BGHM97], [BGHM01] it was adapted to the problem of polynomial equation solving over the *reals*. A successful implementation of the full algorithm ([Lec00]) is based on [GLS01]. A partial implementation of the algorithm (including basic subroutines) is described in [BHMW01] (see also [CHLM00]). So far the account of successive improvements of data structures and algorithms for *symbolic* elimination. The complexity aspect of *numeric* elimination was treated in a series of papers ([SS93a], [SS93b], [SS93c], [SS96], [SS94], [CS99]; see also [BCSS98]). In [CHMP00] the bit complexity aspect of the above mentioned symbolic and numeric algorithms was analyzed and compared. Taking bit complexity and the bit representation of rational numbers into account, it turns out that a numerical adaptation of the above mentioned new symbolic elimination algorithm has the best complexity performance between all known numerical elimination algorithms.

Let us now briefly sketch the known lower bound results for the complexity of arithmetic circuit based elimination procedures. Any elimination algorithm which is *geometrically robust* in the sense of [HMPW98] requires necessarily exponential time on infinitely many inputs. Geometric robustness is a very mild condition that is satisfied by all known (symbolic) elimination procedures.

Moreover, suppose that there is given an elimination algorithm being in a suitable sense "universal", avoiding "unnecessary branchings" and able to compute Zariski closures of constructible sets and "parametric" greatest common divisors of algebraic families of univariate polynomials. Then necessarily this algorithm has to be robust and hence of non–polynomial time complexity [GH01]. In particular, any "reasonable" and "sufficiently general" elimination procedure produces geometrically robust arithmetic circuit representations of suitable elimination polynomials and therefore outputs of non–polynomial size in worst case.

In this paper we are going to argue that the non–polynomial complexity character of the known symbolic *and* numeric geometric elimination procedures is not a special feature of a particular data structure (like the dense, sparse or arithmetic circuit encoding of polynomials), but rather a consequence of the information encoded by the respective data structure.

1.1 Data Structures for Geometric Objects

Informally, we understand by a *data structure* a class, say $\mathcal{D}$, of "simple" mathematical objects which encode another class, say $\mathcal{O}$, of "complicated" ones. An element $D \in \mathcal{D}$ which encodes a mathematical object $O \in \mathcal{O}$ is called a *code* of $\mathcal{O}$. The data structure $\mathcal{D}$ is supposed to be embedded in a context, where we may process its elements, in order to answer a certain catalogue of well defined *questions* about the mathematical objects belonging to $\mathcal{O}$ (or about the object class $\mathcal{O}$ itself). Of course, the choice of the data structure $\mathcal{D}$ depends strongly on the kind of potential questions we are going to ask and on the time we are willing to wait for the answers.

The mathematical objects we are going to consider in this paper will always be polynomial functions or algebraic varieties and their codes will always belong to suitable affine or projective ambient spaces. The size of a code is measured by the dimension of its ambient space.

The (optimal) encoding of *discrete* (e.g. finite) sets of mathematical objects is a well known subject in theoretical computer science and the main theme of Kolmogorov complexity theory ([LV93]; see also [Bor48]).

This paper addresses the problem of optimal encoding of *continuous* classes of mathematical objects. The continuous case differs in many aspects from the discrete one and merits particular attention. Any object class $\mathcal{O}$ we are considering in this paper will possess a *natural topology* and may be thought to be embedded in a (huge) affine or projective ambient space. The given topology of $\mathcal{O}$ becomes always induced by the Zariski (or strong) topology of its ambient space. In this paper, the *closure* $\overline{\mathcal{O}}$ of the object class $\mathcal{O}$ in its ambient space will generally have a natural interpretation as a class of objects of the same nature as $\mathcal{O}$. In this sense we shall interpret a given element of $\overline{\mathcal{O}} \setminus \mathcal{O}$ as a *limit* (or *degenerate*) object of $\mathcal{O}$.

If $\mathcal{O}$ is for example a class of equidimensional closed subvarieties of fixed dimension and degree of a suitable projective space, then the topology and the ambient space of $\mathcal{O}$ may be given by the Chow coordinates of the objects of $\mathcal{O}$. Or if $\mathcal{O}$ is the class of polynomial functions of bounded arithmetic circuit complexity L, then $\mathcal{O}$ is contained in a finite dimensional linear subspace of the corresponding polynomial ring and has therefore a natural topology. The limit objects of $\mathcal{O}$ are then those polynomials which have approximative complexity at most L (see [Ald84] and [BCS97] for details).

A fundamental question addressed in this paper is the following:

given an "efficient" (i.e. short) data structure $\mathcal{D}$ encoding the object class $\mathcal{O}$, how may we find another data structure $\overline{\mathcal{D}}$ encoding the object class $\overline{\mathcal{O}}$? How does the size of $\overline{\mathcal{D}}$ (i.e. the size of its codes) depend on the size of $\mathcal{D}$? We shall see that the answer to this question depends strongly on the further use we wish to make of the data structure $\overline{\mathcal{D}}$.

Let be given a data structure $\mathcal{D}$ encoding an object class $\mathcal{O}$. By assumption, $\mathcal{D}$ is embedded in a suitable affine or projective ambient space from which $\mathcal{D}$ inherits a natural topology. We shall always assume that $\mathcal{D}$ encodes $\mathcal{O}$ *continuously* or *holomorphically*. However, in order to capture the important case of the arithmetic circuit representation of polynomials, we shall not insist on *injective* (*unambiguous*) encodings. Let us now fix a particular object $O \in \mathcal{O}$. For the sake of simplicity, we think for the moment $\mathcal{O}$ and $\mathcal{D}$ equipped with the strong topology. We say that the data structure $\mathcal{D}$ represents the object O *robustly* if for any sequence $(D_i)_{i\in\mathbb{N}}$ of elements of $\mathcal{D}$ encoding a sequence $(O_i)_{i\in\mathbb{N}}$ of objects of $\mathcal{O}$ the following implication holds:

if O is an accumulation point of $(O_i)_{i\in\mathbb{N}}$ then there exists an accumulation point D of $(D_i)_{i\in\mathbb{N}}$ in $\mathcal{D}$ which encodes O.

We say that $\mathcal{D}$ encodes the object O *uniquely* (*injectively*/ *unambiguously*) if $\mathcal{D}$ contains a single element D encoding O. Analogously we say that the data structure $\mathcal{D}$ encodes the object class $\mathcal{O}$ *robustly* (*uniquely*/ *injectively*/ *unambiguously*) if $\mathcal{D}$ encodes robustly (uniquely) each object of $\mathcal{O}$.

In the context of this paper, unambiguous encodings appear as special instances of robust ones and it will turn out that in fact robust encodings are finitely–many–valued encodings.

In algebraic geometry, polynomial equation systems are the "simple" mathematical objects which encode the real objects of interest: algebraic varieties or schemes. Except for the particular case of hypersurfaces, there is no *a priori* priviledged *canonical* equation system that defines a given algebraic variety. It depends on the questions we are going to ask about the given variety, whether we shall feel the need to transform a given equation system into a new, better suited one for answering our questions.

As far as possible, we wish just to modify the syntactical form of our equations, without changing their meaning, represented by the underlying variety or scheme. Let us explain this in two different situations.

Very often the new equation system we are looking for is *uniquely determined* by the underlying variety or scheme and the syntactical requirements the new system has to satisfy. We meet this situation in the particular case of the (reduced) Gröbner basis of a given ideal (representing a scheme) for a previously fixed monomial order. The monomial order we shall choose depends on the kind of questions we are going to ask about the given scheme: we choose an (e.g. lexicographical) elimination order if we wish to "solve" the given equation system (i.e. uncouple its variables) or we choose a graded order if we wish to compute the Hilbert polynomial (the dimension and the degree) of the given (projective) scheme, etc. If we want to analyze a given scheme or variety by means of deformations, suitable (i.e. flat) equation systems, like Gröbner bases, are even mandatory ([BM93]). Although Gröbner bases are able to answer all typical questions about the variety or scheme under consideration, they are not well suited for the less ambitious task of polynomial equation solving (this constitutes the main elimination problem the paper is focusing on).

Since Gröbner bases are able to answer too many questions about the scheme or variety they define, they may become difficult to encode: a complete intersection ideal given by low degree binomial equations, may have a Gröbner basis of doubly exponential degree for a suitable elimination order, whereas it is possible to solve the corresponding elimination problem in singly exponential time using only polynomials of singly exponential degree (see [DFGS91], [KP96], [HMPS00]).

Let us consider another case of this general situation:

for the particular task of polynomial equation solving it suffices to replace the original algebraic variety by a birationally equivalent hypersurface in a suitable ambient space. The (canonical) minimal equation of this hypersurface is the output object of the corresponding elimination procedure. It encodes the necessary information about the dimension and degree of the original algebraic variety, about a maximal set of independent variables and the birational equivalence under consideration. This hypersurface and its minimal equation may be produced by means of generic linear projections (see e.g. [Kro82], [CG83], [GM89], [Can88], [CGH89], [DFGS91], [GH91], [GH93], [KP96], [GHM$^+$98], [GHH$^+$97]) or by means of dual varieties (see [GKZ94] and the references cited there). In the first case, we shall speak about a *Kronecker–like elimination procedure*. However, as a consequence of Bézout's Theorem, the degree of the canonical output equation may increase exponentially if we apply this strategy of elimination.

In either case of this general situation, we need an *input object* (a polynomial equation system describing an algebraic variety or scheme) and an *output object* that describes the same variety or scheme (or a birationally equivalent one) and satisfies some additional syntactical requirements (allowing e.g. the uncoupling of the variables of the original system). The corresponding elimination problem maps input objects to output objects. Since an output object may have degree exponential in the degree of the corresponding input object we are led to ask about short encodings of high degree polynomials in few variables. In this context let us mention the main outcome of [GHM$^+$98], namely the observation that using the arithmetic circuit representation, elimination polynomials (i.e. the output objects of Kronecker–like procedures) have always size polynomial in their degree, whereas the size of their sparse (or dense) representation may become exponential in this quantity. But unfortunately, elimination polynomials may have exponential degrees. This inhibits the elimination procedure of [GHM$^+$98] and [GHH$^+$97] to become polynomial in the input length, at least in worst case.

We consider therefore in more generality the following task:

let be given an elimination problem and a data structure $\mathcal{D}$ encoding the input objects. Find a data structure $\mathcal{D}^*$ encoding the corresponding output objects and an elimination algorithm P which maps input codes belonging to $\mathcal{D}$ to output codes belonging to $\mathcal{D}^*$ and solves the given elimination problem.

1.2 Aim and Scope of Elimination Algorithms

The main question this paper tries to answer can be formulated as follows:

is it possible to find in the given situation a data structure $\mathcal{D}^*$ and an algorithm P such that the size of each output code belonging to $\mathcal{D}^*$ is only *polynomial* in the size of the corresponding input code belonging to $\mathcal{D}$? Under which circumstances do such a data structure $\mathcal{D}^*$ and such an algorithm P exist and under which circumstances do they not?

As mentioned before, the answer depends strongly on the questions about the output objects we want to be answered by the output data structure $\mathcal{D}^*$.

In view of the progress made in [GHM$^+$98], [GHH$^+$97], [GHMP97], [HKP$^+$00], [GLS01] and [HMW01] we limit our attention to basic and relatively simple elimination problems of the following type:

(i) Let be given a zero dimensional algebraic variety V by an input equation system in n variables $X_1, \ldots, X_n$ and given a supplementary input polynomial F in these n variables and possibly some additional parameters $U_1, \ldots, U_m$. Let $X := (X_1, \ldots, X_n)$ and $U := (U_1, \ldots, U_n)$ and let us suppose that the input equation system and F have short encodings in a previously fixed input data structure $\mathcal{D}$. The problem is to find an output data structure $\mathcal{D}^*$ and a Kronecker–like elimination procedure P such that P associates to each input code of $\mathcal{D}$ representing a specialization u of the parameters U of F, an output code of $\mathcal{D}^*$ representing the canonical elimination polynomial of $F(u, X)$ with respect to the given variety V.

(ii) Let be given a class $\mathcal{O}$ of mathematical objects and a data structure $\mathcal{D}$ encoding the object class $\mathcal{O}$. Suppose that $\mathcal{O}$ and $\mathcal{D}$ satisfy all general assumptions we made before on this kind of mathematical entities. Find a data structure $\overline{\mathcal{D}}$ and an algorithm P such that $\overline{\mathcal{D}}$ encodes the topological closure $\overline{\mathcal{O}}$ of the object class $\mathcal{O}$ and such that P maps any element of $\mathcal{D}$ encoding a given object O of $\mathcal{O}$ to an element of $\overline{\mathcal{D}}$ encoding the same object O.

A typical example of such a procedure P for arithmetic circuit represented rational functions of bounded degree d is the "Vermeidung von Divisionen" procedure of [Str73] (see also [KP96]). In this case the limit objects are polynomials of degree bounded by at least $2d + 1$. Another example of such a procedure is the transformation (by means of "tensoring") of approximative algorithms for matrix multiplication into exact ones [BCS97].

All known algorithms solving task (i) or, limited to the context of classical elimination theory, task (ii), possess branching–free versions of the same order of complexity. We will therefore consider only *branching–free algorithms* in this paper. Still more restrictively, we shall always assume that our output codes depend *holomorphically* on our input codes.

Furthermore, the algorithms considered in this paper are supposed to satisfy certain additional requirements. The intuitive meaning of elimination procedure algebraic geometers usually have in mind includes automatically these requirements. We proceed to describe three typical situations and the corresponding requirements:

1. Consider a given elimination problem and suppose that the class of input objects is not intrinsically defined (as e.g. the class of all complete intersection equation systems with polynomials of fixed number of variables and degree), but rather by their encoding in a previously fixed data structure (as e.g. the encoding of the input polynomials by arithmetic circuits of previously bounded size). In this case we have few control about the limit objects of our input object class.

We may now require that our input data structure could be transformed in a new one in order to include also the limit input objects, admitting only a *moderate* growth of the input and output code size for the elimination procedures to be considered (compare task (ii) above).

2. Alternatively, suppose that the output object class is defined as the set of objects produced by some input object. Then we may require that our output data structure could be transformed in a new one able to encode also the limit output objects, admitting again only a moderate growth of output code size.

3. Suppose that we are able to transform the given input object class and its data structure into new ones which represent the same class of algebraic varieties. Suppose furthermore that the code size of the new data structure increases only moderately under this transformation and that the output object class remains unchanged. Since we have modified the input object class we face a new elimination problem with the same semantics as the former one. We may now require to be able to solve the new elimination problem admitting only a moderate growth of output code size.

Of course in all these situations we deal with new algorithms which we suppose again to be branching–free. In a general context this may be impossible. In the limited context of this paper, where we face mainly elimination problems of the above type (i), this assumption does not become too restrictive.

The first two of the above situations refer to the notion of limit object. The reason why we insist to include limit (input and output) objects in our algorithmic considerations is the following:

often it is relatively easy to prove “geometrical theorems” depending on certain conditions of genericity. In many applications of algebraic geometry, these theorems are of few interest since, by lack of information or the large scale of different situations to be faced, there is no reason to believe that the genericity conditions of the theorem under consideration are always satisfied. “True” geometric theorems (like suitable variants of the Dimension Theorem [Lan58] or the Bézout Inequality [Hei79], [Hei83], [Ful84], [Vog84]) do not depend on —hidden or explicit— conditions of genericity and are unrestrictedly applicable. A “true” geometric theorem about a given object class must therefore also hold for the limit objects which represent the “degenerate case”. Such a geometric theorem may for example be the certification of the correctness or the complexity estimate of a given procedure, and this leads us back to the subject of algorithmics.

Let us finish these general observations by a comment about the questions we wish to be answered in this paper by a given data structure $\mathcal{D}$

encoding a given object class $\mathcal{O}$. We shall always require that any element $D \in \mathcal{D}$ encoding an object $O \in \mathcal{O}$ contains enough information in order to distinguish O from other elements of the object class $\mathcal{O}$. Thus a typical question we wish to be answered by the data structure $\mathcal{D}$ is the following:

let D and D' be two elements of $\mathcal{D}$ encoding two objects O and O' of $\mathcal{O}$. Are O and O' identical?

In other words, we require to be able to deduce whether $O = O'$ holds by means of processing the codes D and D'. We call this problem the *identity question* associated to the data structure $\mathcal{D}$. A common way to solve this identity question consists of the transformation of the (supposedly ambiguous) data structure $\mathcal{D}$ in a new one, which encodes the objects of $\mathcal{O}$ injectively.

Another typical question we consider in this paper is the following:

suppose additionally that the object class $\mathcal{O}$ consists of (total) functions which can be evaluated on a continuous or discrete domain R. Suppose furthermore that we have free access to any element of R. Let $D \in \mathcal{D}$ be any code of the given object (i.e. function) $O \in \mathcal{O}$.

The question we wish to be answered by the data structure $\mathcal{D}$ is the following:

for any given argument value $r \in R$, what is the function value $O(r)$?

In other words, we require to be able to compute the function value $O(r)$ by means of processing the code D and the argument value r. We call this problem the *value question* associated to the data structure $\mathcal{D}$. Of course, for a class $\mathcal{O}$ of polynomial functions whose number of variables and degree was previously bounded, the value question for a continuous domain R can be reduced to the value question for a discrete domain or to that of determining, for any monomial M and any polynomial function $O \in \mathcal{O}$, the coefficient of M in the polynomial O.

The value question arises in a natural way in the following typical situation of elimination theory:

let be given, by a suitable encoding, an algebraic input family of n–variate polynomial equation systems. We suppose that the given encoding allows us to decide for each point, say ξ, of the affine n–dimensional space and for each particular input equation system, say E, of the given family, whether ξ is a solution of E (this is typically done by evaluating the n–variate polynomials, which constitute the equation system E, in the point ξ). For the given encoding of such an input system E, we are generally unable to *find* any point ξ of the affine ambient space which solves the system E. A standard task of elimination theory consists now in replacing the given input family of equation systems by a new, semantically equivalent one which still allows to decide whether a given point of the affine ambient space is a solution of a particular equation system of the new family and which has additionally a particular syntactical form which allows, or makes easier, to find solutions of a given equation system (e.g. by means of numerical procedures). In typical situations this task becomes achieved by some Kronecker–like elimination procedure which produces for each n–variate equation system of the input family a single *univariate* output polynomial representing the corresponding output system. In this context, root finding of univariate polynomials is considered as an easier task than solving n–variate equation systems and therefore the output equation system (consisting of a single univariate polynomial) solves our problem for the given input instance. For this reason we require usually that the output data structure produced by the given elimination algorithm answers the corresponding value question (this is quite obvious in case that we are going to apply approximative algorithms for the root finding of univariate polynomials).

1.3 Main Results

The main results of this paper may be paraphrased as follows:

let be given an object class $\mathcal{O}$ and a data structure $\mathcal{D}$ encoding $\mathcal{O}$. Suppose that $\mathcal{O}$ and $\mathcal{D}$ satisfy all assumptions made before (in particular we suppose that $\mathcal{O}$ and $\mathcal{D}$ are contained in suitable affine or projective ambient spaces from which they inherit a natural topology and that the encoding of $\mathcal{O}$ by $\mathcal{D}$ is at least continuous).

(A) Assume that $\mathcal{O}$ is a class of polynomial functions and that its encoding by $\mathcal{D}$ is holomorphic. Suppose furthermore that the elements of $\mathcal{O}$ form a cone if $\mathcal{O}$ has affine ambient space (i.e. we assume that $\mathcal{O}$ is closed under multiplication by scalars). Then there exists a data structure $\overline{\mathcal{D}}$ which encodes the closure class $\overline{\mathcal{O}}$ of $\mathcal{O}$ *injectively* and *robustly*. In particular, $\overline{\mathcal{O}}$ and $\overline{\mathcal{D}}$ are homeomorphic topological spaces. Moreover the size of $\overline{\mathcal{D}}$ (i.e. the dimension of its ambient space) is *linear* in the size of $\mathcal{D}$.

Following (A) we may always replace *efficiently* the given data structure $\mathcal{D}$ by an injective and robust one, say $\overline{\mathcal{D}}$, if we are only interested in a *topological* characterization of the object class $\overline{\mathcal{O}}$ (or $\mathcal{O}$). By means of $\overline{\mathcal{D}}$ we are able to answer efficiently the *identity* question about $\overline{\mathcal{O}}$, but *not necessarily* the *value* question. The assumption that the object class forms a cone in case that $\mathcal{O}$ has affine ambient space is not restrictive in the context of this paper, since $\mathcal{O}$ will be typically a class of functions closed under multiplication by scalars. On the other hand, this assumption guarantees that the encoding of the object class $\overline{\mathcal{O}}$ by the data structure $\overline{\mathcal{D}}$ is not only continuous, but also a closed map.

(B) We exhibit an infinite sequence of elimination problems of the above type (i) such that each particular elimination problem of this sequence has the following complexity behaviour:

the input objects form a class $\mathcal{O}$ of polynomials encoded by a *standard* data structure $\mathcal{D}$ (as e.g. the sparse or the arithmetic circuit representation). The output objects form another class of polynomials, say $\mathcal{O}^*$. Let us consider an *arbitrary* branching–free elimination procedure (in the sense described above) which solves the given elimination problem and which transforms the given data structure $\mathcal{D}$ in a new one, say $\mathcal{D}^*$, encoding the object class $\mathcal{O}^*$. We suppose that the elimination procedure under consideration satisfies the following *robustness condition*:

let $(D_i)_{i\in\mathbb{N}}$ be a sequence of elements of the input data structure $\mathcal{D}$ encoding a sequence $(O_i)_{i\in\mathbb{N}}$ of input objects and let $O \in \mathcal{O}$ be an accumulation point of $(O_i)_{i\in\mathbb{N}}$. Then the elimination procedure under consideration transforms the input code sequence $(D_i)_{i\in\mathbb{N}}$ into a sequence $(D_i^*)_{i\in\mathbb{N}}$ of elements of the output data structure $\mathcal{D}^*$ such that $(D_i^*)_{i\in\mathbb{N}}$ has an accumulation point in $\mathcal{D}^*$ which encodes the output object associated to the given input object O.

From this robustness condition and the particular geometric properties of our elimination problem, we are able to conclude that the size of the data structure $\mathcal{D}^*$ is *necessarily* exponential in the size of $\mathcal{D}$.

In particular if the elimination procedure under consideration maps any two codes of $\mathcal{D}$ representing the same input object $O \in \mathcal{O}$ to the *same* code of the output data structure $\mathcal{D}^*$, or if $\mathcal{D}^*$ encodes *injectively* the output object class $\mathcal{O}^*$, then the above robustness condition will be automatically satisfied.

The limitation to a standard input data structure $\mathcal{D}$ is important in this context. If we could choose the data structure $\mathcal{D}$ freely for the construction of an example, we could easily make the same formal conclusion as in (B) without saying too much about the intrinsic complexity of elimination.

From a purely technical point of view, the example of (B) and the certification of its complexity behaviour is not very surprising. The real importance of this example consists in the formulation of the above robustness condition. This has two aspects. The first aspect is a pragmatic one. Possibly due to cultural reasons, all known universal elimination procedures are robust in the above sense (we call an elimination procedure *universal* is it can be applied to any input system consisting of an arbitrary boolean combination of polynomial equations). The second aspect is a more theoretical one which is the content of our next result.

(C) Suppose that there is given a universal elimination procedure P which is also able to solve in the context of elimination theory suitable problems of the above type (ii). In particular, we suppose that the procedure P is able to eliminate quantifiers in parametric existential first order formulas of the language of the elementary theory of algebraically closed fields of characteristic zero and that P is able to compute the Zariski closure of a given constructible set and the parametric greatest common divisor of two given algebraic families of univariate polynomials. Then, for an appropriate choice of (short encoded) input equations having the same semantics as the examples exhibited in (B), the procedure P may be transformed in a new one, of the same order of complexity, which satisfies the robustness condition of (B). Therefore the universal elimination procedure P cannot be of polynomial (sequential) time complexity.

In conclusion, a universal elimination procedure which is able to solve suitable problems of type (*ii*) cannot be polynomial. Let us remark that all known universal elimination procedures satisfy this requirement since they are based on subroutines (in particular greatest common divisor computations) which behave well under specialization.

Results (A), (B) and (C) are formulated in an *exact* computation model which allows to represent all known symbolic and seminumeric elimination procedures (based on the sparse or dense or the arithmetic circuit representation of polynomials).

Finally we shall consider an appropriate *approximative* computation model which allows to analyze the complexity behaviour of numerical elimination algorithms of the kind of [SS93a], [SS93b], [SS93c], [SS96] and [SS94]. Our last result is the following:

(D) The conclusions of (A) and (B) are also true in the approximative computation model.

The ultimate aim of this paper is not a theoretical but a practical one. We try to give a partial answer to the following fundamental question:

what has to be changed in elimination theory in order to obtain practically efficient algorithms?

We establish a list of implicit or explicit requirements satisfied by all known (symbolic or seminumeric) elimination algorithms. These requirements are: universality, no branchings and robustness for certain simple elimination problems, capacity of computing certain closures (as e.g. equations for the Zariski closure of a given constructible set or the greatest common divisor of two polynomials). Moreover, for suitable inputs, all known universal elimination procedures may be transformed easily into Kronecker–like procedures which are able to evaluate the corresponding canonical elimination polynomial in any given argument or to compute its coefficients. In this sense the known elimination procedures are all able to "compute canonical elimination polynomials".

The fulfillment of these requirements and the capacity of computing canonical elimination polynomials implies the experimentally certified non–

polynomial complexity character of these elimination procedures and explains their practical inefficiency. The results of this paper demonstrate that the complexity problem we focus on is not a question of optimization of algorithms and data structures. There is no way out of the dilemma by changing for example from dense to sparse elimination or to fewnomial theory. Hybridization of symbolic and numeric algorithms leads us again back to the same complexity problems we started from.

The practical outcome of this paper is the conclusion that dramatic complexity improvements of the existing elimination algorithms are only possible by a drastic sacrifice. Essentially we have to renounce to universality or to the computation of canonical elimination polynomials. But what could then be algorithmic elimination? Nothing but a long list of particular procedures which answer particular (mainly discrete, not continuous) questions about particular algebraic varieties.

References

[Ald84] A. Alder. Grenzrang und Grenzkomplexität aus algebraischer und topologischer Sicht Ph. Thesis University of Zürich, 1988.

[BCS97] P. Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, Berlin Heidelberg New York, 1997.

[BCSS98] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer, New York Berlin Heidelberg, 1998.

[BGHM97] B. Bank, M. Giusti, J. Heintz, and G.M. Mbakop. Polar varieties and efficient real equation solving: The hypersurface case. *Journal of Complexity*, 13(1):5–27, 1997.

[BGHM01] B. Bank, M. Giusti, J. Heintz, and G.M. Mbakop. Polar varieties and efficient real elimination. To appear in Math. Zeitschrift, 2001.

[BHMW01] N. Bruno, J. Heintz, G. Matera, and R. Wachenchauzer. Functional programming concepts and straight–line programs in computer algebra. To appear in Mathematics and Computers in Simulation, 2001.

[BM93] D. Bayer and D. Mumford. What can be computed in algebraic geometry ? In D. Eisenbud and L. Robbiano, editors, *Computational Algebraic Geometry and Commutative Algebra*, volume XXXIV of *Symposia Matematica*, pages 1–49. Instituto Nazionale di Alta Matematica, Cambridge University Press, 1993.

[Bor48] E. Borel. La définition en mathématiques. In *Les Grands Courants de la Pensée Mathématique*, pages 24–34. Cahiers du Sud, Paris, 1948.

[Can88] J. Canny. Some algebraic and geometric problems in PSPACE. In *Proceedings 20th. Annual ACM Symposium on Theory of Computing, Chicago, Illinois, 2–4 May 1988*, pages 460–467, New York, 1988. ACM Press.

[CG83] A.L. Chistov and D.Y. Grigoriev. Subexponential time solving systems of algebraic equations. i, ii. LOMI preprints E-9-83, E-10-83, Steklov Institute, Leningrad, 1983.

[CGH89] L. Caniglia, A. Galligo, and J. Heintz. Some new effectivity bounds in computational geometry. In T. Mora et al., editors, *Applied Algebra, Algebraic*

Algorithms and Error Correcting Codes, Proceedings of AAECC-6, volume 357 of *Lecture Notes in Computer Science*, pages 131–152, Berlin Heidelberg New York, 1989. Springer.

[CHLM00] B. Castaño, J. Heintz, J. Llovet, and R. Martínez. On the data structure straight–line program and its implementation in symbolic computation. *Mathematics and Computers in Simulation*, 51:497–528, 2000.

[CHMP00] D. Castro, K. Hägele, J.E. Morais, and L.M. Pardo. Kronecker's and Newton's approaches to solving: a first comparison. *Journal of Complexity*, 17(1):212–303, 2001.

[CS99] F. Cucker and S. Smale. Complexity estimates depending on condition and round–off error. *Journal of the Association for Computing Machinery*, 46(1):113–184, 1999.

[DFGS91] A. Dickenstein, N. Fitchas, M. Giusti, and C. Sessa. The membership problem for unmixed polynomial ideals is solvable in single exponential time. *Discrete Applied Mathematics*, 33:73–94, 1991.

[DH88] J.H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5:29–35, 1988.

[FGM90a] N. Fitchas, A. Galligo, and J. Morgenstern. Algorithmes rapides en sequentiel et en parallèle pour l'élimination des quantificateurs en Géométrie élementaire. In F. Delon, M. Dickmann, and D. Gondard, editors, *Seminaire sur les structures algébriques ordonnées*, volume 32 of *Pub. Math. Univ. Paris VII*, pages 103–145. Paris, 1990.

[FGM90b] N. Fitchas, A. Galligo, and J. Morgenstern. Precise sequential and parallel complexity bounds for quantifier elimination over algebraically closed fields. *Journal of Pure and Applied Algebra*, 67(1):1–14, 1990.

[FGS95] N. Fitchas, M. Giusti, and F. Smietanski. Sur la complexité du théorème des zéros. In J. Guddat et al, editor, *Approximation and Optimization in the Caribbean II, Proceedings 2nd Int. Conf. on Non-Linear Optimization and Approximation*, volume 8 of *Approximation and Optimization*, pages 247–329. Peter Lange Verlag, Frankfurt am Main, 1995.

[Ful84] W. Fulton. *Intersection Theory.* Springer, Berlin Heidelberg New York, 1984.

[GH91] M. Giusti and J. Heintz. Algorithmes - disons rapides - pour la décomposition d' une variété algébrique en composantes irréductibles et équidimensionelles. In T. Mora and C. Traverso, editors, *Effective Methods in Algebraic Geometry, Proceedings of MEGA'90*, volume 94 of *Progress in Mathematics*, pages 169–194, Basel, 1991. Birkhäuser.

[GH93] M. Giusti and J. Heintz. La détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial. In D. Eisenbud and L. Robbiano, editors, *Computational Algebraic Geometry and Commutative Algebra*, volume XXXIV of *Symposia Matematica*, pages 216–256. Cambridge University Press, 1993.

[GH01] M. Giusti and J. Heintz. Kronecker's smart, little black–boxes. To appear in Proceedings of Foundations of Computational Mathematics, Oxford 1999 (FoCM'99), A. Iserles and R. DeVore, editors, Cambridge University Press, 2001.

[GHH+97] M. Giusti, K. Hägele, J. Heintz, J.E. Morais, J.L. Montaña, and L.M. Pardo. Lower bounds for diophantine approximation. *Journal of Pure and Applied Algebra*, 117,118:277–317, 1997.

[GHM+98] M. Giusti, J. Heintz, J.E. Morais, J. Morgenstern, and L.M. Pardo. Straight–line programs in geometric elimination theory. *Journal of Pure and Applied Algebra*, 124:101–146, 1998.

[GHMP95] M. Giusti, J. Heintz, J.E. Morais, and L.M. Pardo. When polynomial equation systems can be solved fast ? In G. Cohen, H. Giusti, and T. Mora, editors, *Applied Algebra, Algebraic Algorithms and Error Correcting Codes, ' Proceedings AAECC-11*, volume 948 of *Lecture Notes in Computer Science*, pages 205–231, Berlin Heidelberg New York, 1995. Springer.

[GHMP97] M. Giusti, J. Heintz, J.E. Morais, and L.M. Pardo. Le rôle des structures de données dans les problèmes d'élimination. *Comptes Rendus de l'Academie de Sciences de Paris*, 325:1223–1228, 1997.

[GHS93] M. Giusti, J. Heintz, and J. Sabia. On the efficiency of effective nullstellensätze. *Computational Complexity*, 3:56–95, 1993.

[GKZ94] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. Birkhäuser, Boston, 1994.

[GLS01] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner free alternative for polynomial system solving. *Journal of Complexity*, 17(1):154–211, 2001.

[GM89] P. Gianni and T. Mora. Algebraic solution of systems of polynomial equations using gröbner bases. In G. Goos and J. Hartmanis, editors, *Applied Algebra, Algebraic Algorithms and Error–Correcting Codes, Proceedings of AAECC–5*, volume 356 of *Lecture Notes in Computer Science*, pages 247–257, Berlin Heidelberg New York, 1989. Springer.

[GS99] M. Giusti and E. Schost. Solving some over–determined systems. In S. Dooley, editor, *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation, ISSAC'99, July 28–31, 1999, Vancouver, Canada*, pages 1–8, New York, 1999. ACM Press.

[GV88] D. Grigoriev and N.N. Vorobjov, Jr. Solving systems of polynomial inequalities in sub–exponential time. *Journal of Symbolic Computation*, 5(1-2):37–64, 1988.

[Hei79] J. Heintz. Definability bounds of first order theories of algebraically closed fields (extended abstract). In L. Buddach, editor, *Proceedings of Fundamentals of Computation Theory, FCT'79, Berlin/ Wendisch- Rietz, 1979*, pages 160–166, Berlin, 1979. Akademie Verlag.

[Hei83] J. Heintz. Definability and fast quantifier elimination in algebraically closed fields. *Theoretical Computer Science*, 24(3):239–277, 1983.

[HKP+00] J. Heintz, T. Krick, S. Puddu, J. Sabia, and A. Waissbein. Deformation techniques for efficient polynomial equation solving. *Journal of Complexity*, 16(1):70–109, 2000.

[HMPS00] K. Hägele, J.E. Morais, L.M. Pardo, and M. Sombra. On the intrinsic complexity of the arithmetic nullstellensatz. *Journal of Pure and Applied Algebra*, 146(2):103–183, 2000.

[HMPW98] J. Heintz, G. Matera, L.M. Pardo, and R. Wachenchauzer. The intrinsic complexity of parametric elimination methods. *Electronic Journal of SADIO*, 1(1):37–51, 1998.

[HMW01] J. Heintz, G. Matera, and A. Waissbein. On the time–space complexity of geometric elimination procedures. *Applicable Algebra in Engineering, Communication and Computing*, 11:239–296, 2001.

[HRS89] J. Heintz, M.-F. Roy, and P. Solernó. On the complexity of semialgebraic sets. In G. Ritter, editor, *Information Processing 89, Proceedings of the IFIP*

11th World Computer Congress, San Francisco, USA, August 28 – September 1, 1989, pages 293–298. North-Holland/IFIP, 1989.

[HRS90] J. Heintz, M.-F. Roy, and P. Solernó. Sur la complexité du principe de tarski-seidenberg. *Bull. Soc. Math. de France*, 118:101–126, 1990.

[HS81] J. Heintz and M. Sieveking. Absolute primality of polynomials is decidable in random polynomial–time in the number of variables. In Shimon Even and Oded Kariv, editors, *ICALP 81: Proceedings 8th International Colloquium on Automata, Languages and Programming, Acre (Akko), Israel, July 13-17, 1981*, volume 115 of *Lecture Notes in Computer Science*, pages 16–28. Springer, 1981.

[HS82] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute. In *International Symposium on Logic and Algorithmic, Zurich 1980*, volume 30 of *Monographie de l'Enseignement Mathématique*, pages 237–254, 1982.

[Ier89] D. Ierardi. Quantifier elimination in the theory of an algebraically closed field. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, Seattle, Washington, 15–17 May 1989*, pages 138–147, New York, 1989. ACM Press.

[Kal88] E. Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *Journal of the Association for Computing Machinery*, 35(1):231–264, 1988.

[KP94] T. Krick and L.M. Pardo. Une approche informatique pour l' approximation diophantienne. *C. R. Acad. Sci. Paris*, 318(1):407–412, 1994.

[KP96] T. Krick and L.M. Pardo. A computational method for diophantine approximation. In L. González-Vega and T. Recio, editors, *Algorithms in Algebraic Geometry and Applications, Proceedings of MEGA'94*, volume 143 of *Progress in Mathematics*, pages 193–254, Basel, 1996. Birkhäuser.

[Kro82] L. Kronecker. Grundzüge einer arithmetischen Theorie de algebraischen Grössen. *J. reine angew. Math.*, 92:1–122, 1882.

[Lan58] S. Lang. *Introduction to Algebraic Geometry.* Interscience, 1958.

[Lec00] G. Lecerf. *Kronecker 0.16beta-2. Reference Manual.* Laboratoire GAGE, École Polytechnique, Palaiseau, France, `http://kronecker.medicis.polytechnique.fr/`, 2000.

[Lec01] G. Lecerf. Computing an equidimensional decomposition of an algebraic variety by means of geometric resolutions. *Proceedings of ISSAC'00 (ACM)*, 209–216, 2001.

[LV93] M. Li and P. Vitányi. *Introduction to Kolmogorov Complexity and its Applications.* Springer, Berlin Heidelberg New York, 1993.

[Mat99] G. Matera. Probabilistic algorithms for geometric elimination. *Applicable Algebra in Engineering, Communication and Computing*, 9(6):463–520, 1999.

[MP93] J.L. Montaña and L.M. Pardo. Lower bounds for arithmetic networks. *Applicable Algebra in Engineering, Communication and Computing*, 4(1):1–24, 1993.

[MP97] B. Mourrain and V. Pan. Solving special polynomial systems by using structural matrices and algebraic residues. In F. Cucker and M. Shub, editors, *Proceedings Foundations of Computational Mathematics (FOCM'97)*, pages 287–304, Berlin Heidelberg New York, 1997. Springer.

[PS98] S. Puddu and J. Sabia. An effective algorithm for quantifier elimination over algebraically closed fields using straight–line programs. *Journal of Pure and Applied Algebra*, 129(2):173–200, 1998.

[Ren92] J. Renegar. On the computational complexity and geometry of the first order theory of the reals. part I: Introduction. preliminaries. the geometry of semi-algebraic sets. the decision problem for the existential theory of the reals. *Journal of Symbolic Computation*, 13(3):255–300, 1992.

[Roj00] J.M. Rojas Computing Complex Dimension Faster and Deterministically (Extended Abstract) arXiv:math.AG/0005028, 2000.

[Sch00] E. Schost. Computing parametric geometric resolutions. Manuscript Ecole Polytechnique, Palaiseau, France, 2000.

[SS93a] M. Shub and S. Smale. Complexity of Bézout's Theorem I: Geometric aspects. *Journal of the AMS*, 6(2):459–501, 1993.

[SS93b] M. Shub and S. Smale. Complexity of Bézout's Theorem II: Volumes and probabilities. In F. Eyssette and A. Galligo, editors, *Computational Algebraic Geometry*, volume 109 of *Progress in Mathematics*, pages 267–285, Basel, 1993. Birkhäuser.

[SS93c] M. Shub and S. Smale. Complexity of Bézout's Theorem III: Condition number and packing. *Journal of Complexity*, 9:4–14, 1993.

[SS94] M. Shub and S. Smale. Complexity of Bézout's Theorem V: Polynomial time. *Theoretical Computer Science*, 133:141–164, 1994.

[SS96] M. Shub and S. Smale. Complexity of Bézout's Theorem IV: Probability of success. *SIAM Journal of Numerical Analysis*, 33:141–164, 1996.

[Str73] V. Strassen. Vermeidung von Divisionen. *Crelle J. Reine Angew. Math,*, 264:182–202, 1973.

[Vog84] W. Vogel. *Results on Bezout's Theorem.* Tata Institute of Fundamental Research. Springer, 1984.

[Wei88] V. Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5:3–27, 1988.