Software Engineering and Complexity in Effective Algebraic Geometry

Non Peer-reviewed author version

Heintz, Joos; KUIJPERS, Bart & Rojas Paredes, Andrés (2011) Software Engineering and Complexity in Effective Algebraic Geometry.

Handle: http://hdl.handle.net/1942/13256

Software Engineering and Complexity in Effective Algebraic Geometry [1]

Joos Heintz[2], Bart Kuijpers[3], Andrés Rojas Paredes[4]

Dedicated to the memory of Jacques Morgenstern
whose ideas inspired this work

October 19, 2011

Abstract

We introduce the notion of a *robust* parameterized arithmetic circuit for the evaluation of algebraic families of multivariate polynomials. Based on this notion, we present a computation model, adapted to Scientific Computing, which captures all known *branching parsimonious* symbolic algorithms in effective Algebraic Geometry. We justify this model by arguments from Software Engineering. Finally we exhibit a class of simple elimination problems of effective Algebraic Geometry which require exponential time to be solved by branching parsimonious algorithms of our computation model.

Keywords: Robust parameterized arithmetic circuit, isoparametric routine, branching parsimonious algorithm, flat family of zero dimensional elimination problems.
MSC: 68Q05, 68Q17, 68Q60, 68N30, 14E99, 14Q99

1 Introduction

We introduce and motivate a practically feasible software architecture based model of branching parsimonious computation using the circuit representation of rational functions as fundamental data type. In this computation model, a routine will accept a circuit as input and produce another circuit as output. Since the basic routines of our computations with circuits will be branching–free and circuits themselves may

[1] Research partially supported by the following Argentinian, Belgian and Spanish grants: CONICET PIP 2461/01, UBACYT 20020100100945, PICT–2010–0525, FWO G.0344.05, MTM2010-16051.

[2] Departamento de Computación, Universidad de Buenos Aires and CONICET, Ciudad Universitaria, Pab. I, 1428 Buenos Aires, Argentina, and Departamento de Matemáticas, Estadística y Computación, Facultad de Ciencias, Universidad de Cantabria, Avda. de los Castros s/n, 39005 Santander, Spain. joos@dc.uba.ar & joos.heintz@unican.es

[3] Database and Theoretical Computer Science Research Group, Hasselt University, Agoralaan, Gebouw D, 3590 Diepenbeek, Belgium. bart.kuijpers@uhasselt.be

[4] Departamento de Computación, Universidad de Buenos Aires, Ciudad Universitaria, Pab. I, 1428 Buenos Aires, Argentina. arojas@dc.uba.ar

be interpreted as computations, the circuits used as data types in our model should be branching–free too. This leads us to introduce and discuss in Section 3.1 the concept of a *parameterized arithmetic circuit*. However, branchings are sometimes unavoidable. But, frequently they may be replaced by limit processes. In order to capture this situation, we shall introduce and discuss the notion of a *robust* parameterized arithmetic circuit.

An important issue will be the concept of *well behavedness* of routines, under certain modifications of the input circuits. This concept will allow us in Section 3.3 to establish our software architecture based model of computation with robust parameterized arithmetic circuits. In this context we shall introduce for our routines the technical notions of well behavedness under *restrictions* and *reductions*, and of *isopametricity* and *coalescence*.

In order to capture the whole spectrum of really existing elimination algorithms in Algebraic Geometry we extend our computational model in Section 3.3.3 admitting some limited branchings. The resulting algorithms are called *branching parsimonious*. Moreover we introduce the concept of a *procedure* as a branching parsimonious algorithm with a particular architecture. Procedures are well suited to discuss computational issues in effective elimination theory. In Section 4, we apply our computation model to this task.

It turns out that already very elementary elimination problems require exponential time to be solved by procedures of our model (see Theorem 10, Proposition 11 and Theorem 12 below). In particular, we exhibit in Section 4.3 a parameterized Boolean circuit whose (standard) arithmetization represents a flat family of zero–dimensional elimination problems which require exponential time to be solved in our model. Moreover, well–known methods based on arithmetization, to count the number of satisfying variable instances of a given Boolean circuit, are of intrinsically exponential complexity character (see Theorem 13 below).

In Section 4.4 we arrive at the conclusion that our method to show lower complexity bounds consists of counting how many steps are necessary to decompose a given rational map into a sequence of "simple" blow ups and a polynomial map.

Finally in Section 4.5 we establish a link between our computation model and our lower bound results with other complexity views in geometric elimination theory. In this context we discuss the BSS–model of [BSS89] and the view of interactive protocols.

Our computation model and complexity results are based on the concept of a *geometrically robust constructible map*. This concept was introduced in [GHMS11] and we develop it further in Section 2, which is devoted to the algebraic geometric underpinning of the present paper.

The relevance of the lower complexity bounds of this paper for elimination problems depends on the "naturalness" of the computation model. Therefore we emphasize throughout this article the arguments which justify our computation model. Of course, these arguments cannot be entirely of mathematical nature. In this paper they are borrowed from Software Engineering which constitutes a discipline which analyzes and qualifies practical programming issues. In these terms we show that a circuit based algorithm which solves most elementary parametric elimination problems and which is programmed under the application of the most common rules of Software Engineering, can never be efficient.

2 Concepts and tools from Algebraic Geometry

In this section, we use freely standard notions and notations from Commutative Algebra and Algebraic Geometry. These can be found for example in [Lan93], [ZS60], [Kun85] and [Sha94]. In Sections 2.2 and 2.3, we introduce the notions and definitions which constitute our fundamental tool for the modelling of elimination problems and algorithms. Most of these notions and their definitions are taken from [GHMS11].

2.1 Basic notions and notations

For any $n \in \mathbb{N}$, we denote by $\mathbb{A}^n := \mathbb{A}^n(\mathbb{C})$ the n–dimensional affine space $\mathbb{C}^n$ equipped with its respective Zariski and Euclidean topologies over $\mathbb{C}$. In algebraic geometry, the Euclidean topology of $\mathbb{A}^n$ is also called the *strong topology.* We shall use this terminology only exceptionally.

Let $X_1, \ldots, X_n$ be indeterminates over $\mathbb{C}$ and let $X := (X_1, \ldots, X_n)$. We denote by $\mathbb{C}[X]$ the ring of polynomials in the variables X with complex coefficients.

Let V be a closed affine subvariety of $\mathbb{A}^n$, that is, the set of common zeros in $\mathbb{A}^n$ of a finite set of polynomials belonging to $\mathbb{C}[X]$. As usual, we write $\dim V$ for the dimension of the variety V. Let $C_1, \ldots, C_s$ be the irreducible components of V. For $1 \leq j \leq s$ we define the degree of C_j as the number of points which arise when we intersect C_j with $\dim C_j$ many generic affine hyperplanes of $\mathbb{A}^n$. Observe that this number is a well–determined positive integer which we denote by $\deg C_j$. The *(geometric) degree* $\deg V$ of V is defined by $\deg V := \sum_{1 \leq j \leq s} \deg C_j$. This notion of degree satisfies the so called Bezout Inequality. Namely, for another closed affine subvariety W of $\mathbb{A}^n$ we have $\deg V \cap W \leq \deg V \cdot \deg W$.

For details we refer to [Hei83], where the notion of geometric degree was introduced and the Bezout Inequality was proved for the first time (other references are [Ful84] and [Vog84]).

For $f_1, \ldots, f_s, g \in \mathbb{C}[X]$ we shall use the notation $\{f_1 = 0, \ldots, f_s = 0\}$ in order to denote the closed affine subvariety V of $\mathbb{A}^n$ defined by $f_1, \ldots, f_s$ and the notation $\{f_1 = 0, \ldots, f_s = 0, g \neq 0\}$ in order to denote the Zariski open subset V_g of V defined by the intersection of V with the complement of $\{g = 0\}$. Observe that V_g is a locally closed affine subvariety of $\mathbb{A}^n$ whose coordinate ring is the localization $\mathbb{C}[V]_g$ of $\mathbb{C}[V]$.

We denote by $I(V) := \{f \in \mathbb{C}[X] : f(x) = 0 \text{ for any } x \in V\}$ the ideal of definition of V in $\mathbb{C}[X]$ and by $\mathbb{C}[V] := \{\varphi : V \to \mathbb{C} \ ; \text{ there exists } f \in \mathbb{C}[X] \text{ with } \varphi(x) = f(x) \text{ for any } x \in V\}$ its coordinate ring. Observe that $\mathbb{C}[V]$ is isomorphic to the quotient $\mathbb{C}$–algebra $\mathbb{C}[V] := \mathbb{C}[X]/I(V)$. If V is irreducible, then $\mathbb{C}[V]$ is zero–divisor free and we denote by $\mathbb{C}(V)$ the field formed by the rational functions of V with maximal domain ($\mathbb{C}(V)$ is called the rational function field of V). Observe that $\mathbb{C}(V)$ is isomorphic to the fraction field of the integral domain $\mathbb{C}[V]$.

In the general situation where V is an arbitrary closed affine subvariety of $\mathbb{A}^n$, the notion of a rational function of V has also a precise meaning. The only point to underline is that the domain, say U, of a rational function of V has to be a maximal Zariski open and dense subset of V to which the given rational function can be extended. In particular, U has a nonempty intersection with any of the irreducible components of V.

We denote by $\mathbb{C}(V)$ the $\mathbb{C}$–algebra formed by the rational functions of V. In algebraic terms, $\mathbb{C}(V)$ is the total quotient ring of $\mathbb{C}[V]$ and is isomorphic to the direct product of the rational function fields of the irreducible components of V.

Let be given a partial map $\phi : V \dashrightarrow W$, where V and W are closed subvarieties of some affine spaces $\mathbb{A}^n$ and $\mathbb{A}^m$, and let $\phi_1, \ldots, \phi_m$ be the components of ϕ. With these notations we have the following definitions which can be found in [GHMS11]:

Definition 1 (Polynomial map) *The map ϕ is called a morphism of affine varieties or just polynomial map if the complex valued functions $\phi_1, \ldots, \phi_m$ belong to $\mathbb{C}[V]$. Thus, in particular, ϕ is a total map.*

Definition 2 (Rational map) *We call ϕ a rational map of V to W, if the domain U of ϕ is a Zariski open and dense subset of V and $\phi_1, \ldots, \phi_m$ are the restrictions of suitable rational functions of V to U.*

Observe that our definition of a rational map differs from the usual one in Algebraic Geometry, since we do not require that the domain U of ϕ is maximal. Hence, in the case $m := 1$, our concepts of rational function and rational map do not coincide.

2.1.1 Constructible sets and constructible maps

Let $\mathcal{M}$ be a subset of some affine space $\mathbb{A}^n$ and, for a given nonnegative integer m, let $\phi : \mathcal{M} \dashrightarrow \mathbb{A}^m$ be a partial map.

Definition 3 (Constructible set) *We call the set $\mathcal{M}$* constructible *if $\mathcal{M}$ is definable by a Boolean combination of polynomial equations.*

A basic fact we shall use in the sequel is that if $\mathcal{M}$ is constructible, then its Zariski closure is equal to its Euclidean closure (see, e.g., [Mum88], Chapter I, §10, Corollary 1). In the same vein we have the following definition.

Definition 4 (Constructible map) *We call the partial map ϕ* constructible *if the graph of ϕ is constructible as a subset of the affine space $\mathbb{A}^n \times \mathbb{A}^m$.*

We say that ϕ is *polynomial* if ϕ is the restriction of a morphism of affine varieties $\mathbb{A}^n \to \mathbb{A}^m$ to a constructible subset $\mathcal{M}$ of $\mathbb{A}^n$ and hence a total map from $\mathcal{M}$ to $\mathbb{A}^m$. Furthermore, we call ϕ a *rational* map of $\mathcal{M}$ if the domain U of ϕ is contained in $\mathcal{M}$ and ϕ is the restriction to $\mathcal{M}$ of a rational map of the Zariski closure $\overline{\mathcal{M}}$ of $\mathcal{M}$. In this case U is a Zariski open and dense subset of $\mathcal{M}$.

Since the elementary, i.e., first–order theory of algebraically closed fields with constants in $\mathbb{C}$ admits quantifier elimination, constructibility means just elementary definability. In particular, ϕ is constructible implies that the domain and the image of ϕ are constructible subsets of $\mathbb{A}^n$ and $\mathbb{A}^m$, respectively.

Remark 1 *A partial map $\phi : \mathcal{M} \dashrightarrow \mathbb{A}^m$ is constructible if and only if it is piecewise rational. If ϕ is constructible there exists a Zariski open and dense subset U of $\mathcal{M}$ such that the restriction $\phi|_U$ of ϕ to U is a rational map.*

For details we refer to [GHMS11], Lemma 1.

2.2 Weakly continuous, strongly continuous, topologically robust and hereditary maps

We are now going to present the notions of a weakly continuous, a strongly continuous, a topologically robust, a geometrically robust and a hereditary map of a constructible set $\mathcal{M}$. These five notions will constitute our fundamental tool for the modelling of elimination problems and algorithms.

Definition 5 *Let $\mathcal{M}$ be a constructible subset of $\mathbb{A}^n$ and let $\phi : \mathcal{M} \to \mathbb{A}^m$ be a (total) constructible map. We consider the following four conditions:*

(i) there exists a Zariski open and dense subset U of $\mathcal{M}$ such that the restriction $\phi|_U$ of ϕ to U is a rational map of $\mathcal{M}$ and the graph of ϕ is contained in the Zariski closure of the graph of $\phi|_U$ in $\mathcal{M} \times \mathbb{A}^m$;

(ii) ϕ is continuous with respect to the Euclidean, i.e., strong, topologies of $\mathcal{M}$ and $\mathbb{A}^m$;

(iii) for any sequence $(x_k)_{k\in\mathbb{N}}$ of points of $\mathcal{M}$ which converges in the Euclidean topology to a point of $\mathcal{M}$, the sequence $(\phi(x_k))_{k\in\mathbb{N}}$ is bounded;

(iv) for any constructible subset $\mathcal{N}$ of $\mathcal{M}$ the restriction $\phi|_{\mathcal{N}} : \mathcal{N} \to \mathbb{A}^m$ is an extension of a rational map of $\mathcal{N}$ and the graph of $\phi|_{\mathcal{N}}$ is contained in the Zariski closure of this rational map in $\mathcal{N} \times \mathbb{A}^m$.

We call the map ϕ

- **weakly continuous** *if ϕ satisfies condition (i),*
- **strongly continuous** *if ϕ satisfies condition (ii),*

- **topologically robust** *if ϕ satisfies conditions* (*i*) *and* (*iii*),
- **hereditary** *if ϕ satisfies condition* (*iv*).

In all these cases we shall refer to $\mathcal{M}$ as the domain of definition of ϕ or we shall say that ϕ is defined on $\mathcal{M}$.

Remark 2 *A strongly continuous constructible map is always weakly continuous, topologically robust and hereditary.*

For details we refer to [GHMS11], Lemma 4.

The concept of hereditarity sounds rather abstract and axiomatic. We shall need it in Section 3 for a mathematically correct and complete formulation of our computation model. In Section 2.3, we shall establish an algebraic condition, namely geometric robustness, which implies hereditarity.

2.3 The concept of robustness for constructible maps

In this Section we introduce the algebraic–geometric tools we shall use in Section 3 and 4 for the mathematical modelling of algorithms which solve parameterized computational problems. The main issue of this section will be the notion of a *geometrically robust constructible map* which captures simultaneously the concepts of topological robustness and hereditarity introduced in Section 2.2

We first characterize in algebraic terms the concept of topological robustness (Theorem 3 below). In Section 3 we shall interpret topological robustness as *coalescence*, an informal concept whose exact definition depends on the context. For example in Interpolation theory coalescence refers to certain types of "convergence" of problems and algorithms (see [BC97], [dBR92], [Olv06] and [GHMS11] for details). In this paper coalescence will be the algorithmic counterpart of topological robustness.

Finally, we introduce the notion of a geometrically robust constructible map and show that such maps are always hereditary. In particular they are topologically robust and give rise to coalescent algorithms.

2.3.1 An algebraic characterization of the notion of topological robustness

In this subsection, we present an algebraic–geometric result of [GHMS11] which will be relevant in Sections 2.3.2, 3 and 4.

For the moment let us fix a constructible subset $\mathcal{M}$ of the affine space $\mathbb{A}^n$ and a (total) constructible map $\phi : \mathcal{M} \to \mathbb{A}^m$ with components $\phi_1, \dots, \phi_m$. Suppose the map ϕ is weakly continuous in the sense of Definition 5 in Section 2.2.

We consider now the Zariski closure $\overline{\mathcal{M}}$ of the constructible subset $\mathcal{M}$ of $\mathbb{A}^n$. Observe that $\overline{\mathcal{M}}$ is a closed affine subvariety of $\mathbb{A}^n$ and that we may interpret $\mathbb{C}(\overline{\mathcal{M}})$ as a $\mathbb{C}[\overline{\mathcal{M}}]$–module (or $\mathbb{C}[\overline{\mathcal{M}}]$–algebra).

Fix now an arbitrary point x of $\overline{\mathcal{M}}$.

By $\mathfrak{M}_x$ we denote the maximal ideal of coordinate functions of $\mathbb{C}[\overline{\mathcal{M}}]$ which vanish at the point x.

By $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$ we denote the local $\mathbb{C}$–algebra of the variety $\overline{\mathcal{M}}$ at the point x, i.e., the localization of $\mathbb{C}[\overline{\mathcal{M}}]$ at the maximal ideal $\mathfrak{M}_x$.

By $\mathbb{C}(\overline{\mathcal{M}})_{\mathfrak{M}_x}$ we denote the localization of the $\mathbb{C}[\overline{\mathcal{M}}]$–module $\mathbb{C}(\overline{\mathcal{M}})$ at $\mathfrak{M}_x$.

The constructible map ϕ is by assumption weakly continuous. Hence we may interpret $\phi_1, \dots, \phi_m$ as rational functions of the affine variety $\overline{\mathcal{M}}$ and therefore as elements of the total fraction ring $\mathbb{C}(\overline{\mathcal{M}})$ of $\mathbb{C}[\overline{\mathcal{M}}]$.

Thus $\mathbb{C}[\overline{\mathcal{M}}][\phi_1, \dots, \phi_m]$ and $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \dots, \phi_m]$ are $\mathbb{C}$–subalgebras of $\mathbb{C}(\overline{\mathcal{M}})$ and $\mathbb{C}(\overline{\mathcal{M}})_{\mathfrak{M}_x}$ which contain $\mathbb{C}[\overline{\mathcal{M}}]$ and $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$, respectively.

With these notations we are able to formulate the following statement which establishes the bridge to an algebraic understanding of the notion of topological robustness.

Theorem 3 *([GHMS11], Corollary 11) Let notations and assumptions be as before and suppose that the constructible map $\phi : \mathcal{M} \to \mathbb{A}^m$ is weakly continuous. Then ϕ is topologically robust if and only if for any point x of $\mathcal{M}$ the $\mathbb{C}$–algebra $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \dots, \phi_m]$ is a finite $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$–module.*

The only if part of Theorem 3 is an almost immediate consequence of [CGH$^+$03], Lemma 3, which in its turn is based on a non–elementary and deep result from Algebraic Geometry, namely Zariski's Main Theorem (see, e.g., [Ive73], §IV.2). Theorem 3 and Theorem 5 below will be omnipresent in Sections 3 and 4. They contribute to establish a well–founded link between Computer Science and Algebraic Geometry.

Let $\phi : \mathcal{M} \to \mathbb{A}^m$ be a topologically robust constructible map and let u be an arbitrary point of $\mathcal{M}$. From Theorem 3 one deduces easily that for all sequences $(u_k)_{k \in \mathbb{N}}$ of points $u_k \in \mathcal{M}$ which converge to u, the sequences $(\phi(u_k))_{k \in \mathbb{N}}$ have only finitely many distinct accumulation points.

2.3.2 The notion of geometrical robustness

The main mathematical tool of Section 3 of this paper is the notion of geometrical robustness we are going to introduce now. We shall use the same notations as in Section 2.3.1.

Definition 6 *Let $\mathcal{M}$ be a constructible subset of a suitable affine space and let $\phi : \mathcal{M} \to \mathbb{A}^m$ be a (total) constructible map with components $\phi_1, \dots, \phi_m$. According to Remark 1 we may interpret $\phi_1, \dots, \phi_m$ as rational maps of $\overline{\mathcal{M}}$. We call ϕ geometrically robust if for any point $x \in \mathcal{M}$ the following two conditions are satisfied:*

(i) $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \dots, \phi_m]$ is a finite $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$–module.

(ii) $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \dots, \phi_m]$ is a local $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$–algebra whose maximal ideal is generated by $\mathfrak{M}_x$ and $\phi_1 - \phi_1(x), \dots, \phi_m - \phi_m(x)$.

Observe that the notion of a geometrically robust map makes also sense when $\mathbb{C}$ is replaced by an arbitrary algebraically closed field (of any characteristic). In view of Theorem 3 the same is true for the notion of a topologically robust map. In this sense we have the following fundamental result.

Proposition 4 *Geometrically robust constructible maps are weakly continuous, hereditary and in particular topologically robust. If we restrict a geometrically robust constructible map to a constructible subset of its domain of definition we obtain again a geometrically robust map. Moreover the composition and the cartesian product of two geometrically robust constructible maps are geometrically robust. The geometrically robust constructible functions form a commutative* $\mathbb{C}$*–algebra which contains the polynomial functions.*

We are not going to prove Proposition 4 here. Weak continuity and hereditarity of geometrically robust constructible maps with *irreducible* domains of definition is the content of [GHMS11], Proposition 16, Theorem 17 and Corollary 18. These results imply also that restrictions of such maps to irreducible constructible subsets of their domains of definition are again geometrically robust. From this one deduces immediately the same statements for the case of arbitrary domains of definition. Topological robustness follows from Theorem 3 above. Closedness under composition is a consequence of the transitivity law for integral dependence. One infers from Definition 6 closedness under cartesian products and that the geometrically robust constructible functions form a commutative $\mathbb{C}$–algebra which contains the polynomial functions. The alluded proofs work over arbitrary algebraically closed fields.

In this paper we shall restrict our attention to the algebraically closed field $\mathbb{C}$. In this particular case we have the following characterization of geometrically robust constructible maps.

Theorem 5 *Let assumptions and notations be as before. Then the constructible map* $\phi : \mathcal{M} \to \mathbb{A}^m$ *is geometrically robust if and only if* ϕ *is strongly continuous.*

Proof. Suppose that the constructible map ϕ is geometrically robust. We are first going to show that ϕ is weakly continuous.

By Remark 1 there exists a Zariski open and dense subset U of $\mathcal{M}$ such that the restriction map $\phi|_U$ is rational. Let $Y_1, \dots, Y_m$ be new indeterminates, $Y := (Y_1, \dots, Y_m)$ and suppose that the affine ambient space of $\mathcal{M}$ has dimension n. Observe that any $(n+m)$–variate polynomial over $\mathbb{C}$ which vanishes on the graph of the rational map $\phi|_U$ gives rise to a polynomial $A \in \mathbb{C}[\overline{\mathcal{M}}][Y]$ with $A[\phi_1, \dots, \phi_m] = 0$.

Let x be an arbitrary point of $\mathcal{M}$ and consider A as an element of $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[Y]$. Denote by $A(x, \phi(x))$ the value of A at $(x, \phi(x))$. Then condition (ii) of Definition 6 implies that $A[\phi_1, \dots, \phi_m] - A(x, \phi(x))$ belongs to the maximal ideal of $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \dots, \phi_m]$. From $A[\phi_1, \dots, \phi_m] = 0$ we deduce now $A(x, \phi(x)) = 0$.

Since the choice of $x \in \mathcal{M}$ was arbitrary, we conclude that A vanishes on the graph of ϕ. This implies that the graph of ϕ is contained in the Zariski closure of the graph of $\phi|_U$. Hence ϕ is weakly continuous.

Let be given an arbitrary point $x \in \mathcal{M}$ and a sequence $(x_k)_{k \in \mathbb{N}}$, $x_k \in \mathcal{M}$, which converges to x in the strong topology of $\mathcal{M}$. We are now going to show that the sequence $(\phi(x_k))_{k \in \mathbb{N}}$ converges to $\phi(x)$.

Since ϕ is weakly continuous, we deduce from condition (i) of Definition 6 and Theorem 3 that the sequence $(\phi(x_n))_{k \in \mathbb{N}}$ contains at least one accumulation point, say $a = (a_1, \dots, a_m)$, which belongs to $\mathbb{A}^m$. Let $\mathfrak{a}$ be the ideal of all polynomials $A \in \mathbb{C}[\mathcal{M}]_{\mathfrak{M}_x}[Y]$ that vanish at the point $(x, a) \in \mathbb{A}^n \times \mathbb{A}^m$. Without loss of generality we may assume that the sequence $(\phi(x_k))_{k \in \mathbb{N}}$ converges to a. Let $\widetilde{\mathfrak{a}} := \{A(\phi); A \in \mathfrak{a}\}$ be the image of the ideal $\mathfrak{a}$ under the surjective $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$–algebra homomorphism $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[Y] \to \mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \dots, \phi_m]$ which maps $Y_1, \dots, Y_m$ onto $\phi_1, \dots, \phi_m$. Observe that $\widetilde{\mathfrak{a}}$ is an ideal of $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \dots, \phi_m]$.

We are now going to show the following statement.

Claim 6 *The ideal $\widetilde{\mathfrak{a}}$ is proper.*

Proof of the claim. Suppose that the ideal $\widetilde{\mathfrak{a}}$ is not proper. Then there exists a polynomial $A = \sum_{j_1, \dots, j_m} a_{j_1 \cdots j_m} Y_1^{j_1} \dots Y_m^{j_m}$ of $\mathfrak{a}$, with $a_{j_1 \cdots j_m} \in \mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$, which satisfies the condition $\sum_{j_1, \dots, j_m} a_{j_1 \cdots j_m} \phi_1^{j_1} \dots \phi_m^{j_m} = A(\phi) = 1$. Since for any m–tuple of indices $j_1, \dots, j_m$ the rational function $a_{j_1 \cdots j_m}$ of $\overline{\mathcal{M}}$ is defined at x and the sequence $(x_k)_{k \in \mathbb{N}}$ converges to x, we may assume without loss of generality that $a_{j_1 \cdots j_m}$ is defined at x_k for any $k \in \mathbb{N}$ and that $(a_{j_1 \cdots j_m}(x_k))_{k \in \mathbb{N}}$ converges to $a_{j_1 \cdots j_m}(x)$. We may therefore write $A^{(x')} := \sum a_{j_1 \cdots j_m}(x') Y_1^{j_1} \dots Y_m^{j_m} \in \mathbb{C}[Y]$ for $x' := x$ or $x' := x_k$, $k \in \mathbb{N}$. From $A \in \mathfrak{a}$ we deduce $A^{(x)}(a) = 0$. By assumption $(\phi(x_k))_{k \in \mathbb{N}}$ converges to a. Hence the sequence of complex numbers $(A^{(x_k)}(\phi(x_k)))_{k \in \mathbb{N}}$ converges to $A^{(x)}(a) = 0$. On the other hand $A(\phi) = 1$ and the weak continuity of ϕ imply $A^{(x_k)}(\phi(x_k)) = 1$ for any $k \in \mathbb{N}$. This contradiction proves our claim.

From condition (ii) of Definition 6 we deduce that the $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$–algebra $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \dots, \phi_m]$ contains a single maximal ideal, say $\mathfrak{M}$, and that $\mathfrak{M}$ is generated by $\mathfrak{M}_x$ and $\phi_1 - \phi_1(x), \dots, \phi_m - \phi_m(x)$.

Since by Claim 6 the ideal $\widetilde{\mathfrak{a}}$ is proper, $\widetilde{\mathfrak{a}}$ must be contained in $\mathfrak{M}$. Observe that the polynomials $Y_1 - a_1, \dots, Y_m - a_m$ belong to $\mathfrak{a}$. Hence $\phi_1 - a_1, \dots, \phi_m - a_m$ belong to $\widetilde{\mathfrak{a}}$ and therefore also to $\mathfrak{M}$. Since $\mathfrak{M}$ is proper, this is only possible if $a_1 = \phi_1(x), \dots, a_m = \phi_m(x)$ holds.

Thus the sequence $(\phi(x_k))_{k \in \mathbb{N}}$ converges to $\phi(x)$.

Suppose now that the constructible map ϕ is strongly continuous. From Remark 2 we deduce that ϕ is topologically robust. Theorem 3 implies now that ϕ satisfies condition (i) of Definition 6 at any point of $\mathcal{M}$.

Let x be an arbitrary point of $\mathcal{M}$. We have to show that ϕ satisfies at x condition (ii) of Definition 6.

Since the graph of ϕ is constructible, its strong and Zariski closures in $\mathcal{M} \times \mathbb{A}^m$ coincide. Moreover, since ϕ is by assumption strongly continuous, its graph is closed with respect to the strong topology of $\mathcal{M} \times \mathbb{A}^m$ and therefore also with respect to the Zariski topology. Let $\mathfrak{a}$ be an arbitrary maximal ideal of the $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$–algebra $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\phi_1, \ldots, \phi_m]$. Then there exists a point $a = (a_1, \ldots, a_m)$ of $\mathbb{A}^m$ such that $\mathfrak{a}$ is generated by $\mathfrak{M}_x$ and $\phi_1 - a_1, \ldots, \phi_m - a_m$. Thus $(x, a) \in \mathcal{M} \times \mathbb{A}^m$ belongs to the Zariski closure of the graph of ϕ in $\mathcal{M} \times \mathbb{A}^m$ and therefore to the graph of ϕ itself. This implies $a = \phi(x)$. With other words, $\mathfrak{a}$ is generated by $\mathfrak{M}_x$ and $\phi_1 - \phi_1(x), \ldots, \phi_m - \phi_m(x)$. There is exactly one ideal of $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\varphi_1, \ldots, \varphi_m]$ which satisfies this condition. Therefore the $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}$–algebra $\mathbb{C}[\overline{\mathcal{M}}]_{\mathfrak{M}_x}[\varphi_1, \ldots, \varphi_m]$ is local and condition (ii) and Definition 6 is satisfied at the point $x \in \mathcal{M}$. ■

Theorem 5 is new. It gives a topological motivation for the rather abstract, algebraic notion of geometrical robustness. The reader not acquainted with commutative algebra may just identify the concept of geometrical robustness with that of strong continuity of constructible maps.

Observe that Proposition 4 follows immediately from Theorem 5 in the case of the algebraically closed field $\mathbb{C}$.

The origin of the concept of a geometrically robust map can be found, implicitly, in [GH01]. It was introduced explicitly for constructible maps with irreducible domains of definition in [GHMS11], where it is used to analyze the complexity character of multivariate Hermite–Lagrange interpolation. The concept of a geometrically robust map is therefore well motivated from the point of view of Computer Science.

3 A software architecture based model for computations with parameterized arithmetic circuits

3.1 Parameterized arithmetic circuits and their semantics

The routines of our computation model, which will be introduced in Section 3.3, operate with circuits representing parameter dependent rational functions. They will behave well under restrictions. In this spirit, the objects of our abstract data types will be parameter dependent multivariate rational functions over $\mathbb{C}$, the concrete objects of our classes will be parameterized arithmetic circuits and our abstraction function will associate circuits with rational functions. In what follows, $\mathbb{C}$ may always be replaced, mutatis mutandis, by an arbitrary algebraically closed field (of any characteristic).

Let us fix natural numbers n and r, indeterminates $X_1, \ldots, X_n$ and a non–empty constructible subset $\mathcal{M}$ of $\mathbb{A}^r$. By $\pi_1, \ldots, \pi_r$ we denote the restrictions to $\mathcal{M}$ of the canonical projections $\mathbb{A}^r \to \mathbb{A}^1$.

A *(by $\mathcal{M}$) parameterized arithmetic circuit* β (with *basic parameters* $\pi_1, \ldots, \pi_r$ and *inputs* $X_1, \ldots, X_n$) is a labelled directed acyclic graph (labelled DAG) satisfying the following conditions:
each node of indegree zero is labelled by a scalar from $\mathbb{C}$, a basic parameter $\pi_1, \ldots, \pi_r$ or a input variable $X_1, \ldots, X_n$. Following the case, we shall refer to the *scalar, basic parameter* and (standard) *input* nodes of β. All other nodes of β have indegree two and are called *internal.* They are labelled by arithmetic operations (addition, subtraction, multiplication, division). A *parameter* node of β depends only on scalar and basic parameter nodes, but not on any input node of β. An addition or multiplication node whose two ingoing edges depend on an input is called *essential.* The same terminology is applied to division nodes whose second argument depends on an input. Moreover, at least one circuit node becomes labelled as output. Without loss of generality we may suppose that all nodes of outdegree zero are outputs of β.

We consider β as a syntactical object which we wish to equip with a certain semantics. In principle there exists a canonical evaluation procedure of β assigning to each node a rational function of $\mathcal{M} \times \mathbb{A}^n$ which, in case of a parameter node, may also be interpreted as a rational function of $\mathcal{M}$. We call such a rational function an *intermediate result* of β.

The evaluation procedure may fail if we divide at some node an intermediate result by another one which vanishes on a Zariski dense subset of a whole irreducible component of $\mathcal{M} \times \mathbb{A}^n$. If this occurs, we call the labelled DAG β *inconsistent*, otherwise *consistent.* From [CGH$^+$03], Corollary 2 (compare also [HS80], Theorem 4.4 and [GH01], Lemma 3) one deduces easily that testing whether an intermediate result of β vanishes on a Zariski dense subset of a whole irreducible component of $\mathcal{M} \times \mathbb{A}^n$ can efficiently be reduced to the same task for circuit represented rational functions of $\mathcal{M}$ (the procedure is of non–uniform deterministic or alternatively of uniform–probabilistic nature).

Mutatis mutandis the same is true for identity checking between intermediate results of β. If $\mathcal{M}$ is irreducible, both tasks boil down to an identity–to–zero test on $\mathcal{M}$. In case that $\mathcal{M}$ is not Zariski dense in $\mathbb{A}^r$, this issue presents a major open problem in modern Theoretical Computer Science (see [Sax09] and [Shp10] for details).

If nothing else is said, we shall from now on assume that β is a consistent parameterized arithmetic circuit. The intermediate results associated with output nodes will be called *final results* of β.

We call an intermediate result associated with a parameter node a *parameter* of β and interpret it generally as a rational function of $\mathcal{M}$. A parameter associated with a node which has an outgoing edge into a node which depends on some input of β is called *essential.* In the sequel we shall refer to the constructible set $\mathcal{M}$ as the *parameter domain* of β.

We consider β as a syntactic object which represents the final results of β, i.e., the rational functions of $\mathcal{M} \times \mathbb{A}^n$ assigned to its output nodes. In this way becomes introduced an abstraction function which associates β with these rational functions. This abstraction function assigns therefore to β a rational map $\mathcal{M} \times \mathbb{A}^n \dashrightarrow \mathbb{A}^q$, where q is the number of output nodes of β. On its turn, this rational map may also be understood as a (by $\mathcal{M}$) parameterized family of rational maps $\mathbb{A}^n \dashrightarrow \mathbb{A}^q$.

Now we suppose that the parameterized arithmetic circuit β has been equipped with an additional structure, linked to the semantics of β. We assume that for each node ρ of β there is given a *total* constructible map $\mathcal{M} \times \mathbb{A}^n \to \mathbb{A}^1$ which extends the intermediate result associated with ρ. In this way, if β has K nodes, we obtain a total constructible map $\Omega : \mathcal{M} \times \mathbb{A}^n \to \mathbb{A}^K$ which extends the rational map $\mathcal{M} \times \mathbb{A}^n \dashrightarrow \mathbb{A}^K$ given by the labels at the indegree zero nodes and the intermediate results of β.

Definition 7 (Robust circuit) *Let notations and assumptions be as before. The pair (β, Ω) is called a robust parameterized arithmetic circuit if the constructible map Ω is geometrically robust.*

We shall make the following two observations to this definition.

We state our first observation. Suppose that (β, Ω) is robust. Then the constructible map $\Omega : \mathcal{M} \times \mathbb{A}^n \to \mathbb{A}^K$ is geometrically and hence also topologically robust and hereditary. Moreover, there exists at most one geometrically robust constructible map $\Omega : \mathcal{M} \times \mathbb{A}^n \to \mathbb{A}^K$ which extends the rational map $\mathcal{M} \times \mathbb{A}^n \dashrightarrow \mathbb{A}^K$ introduced before. Therefore we shall apply from now on the term “robust” also to the circuit β.

Let us now state our second observation. We may consider the parameterized circuit β as a program which solves the problem to evaluate, for any sufficiently generic parameter instance $u \in \mathcal{M}$, the rational map $\mathbb{A}^n \dashrightarrow \mathbb{A}^q$ which we obtain by specializing to the point u the first argument of the rational map $\mathcal{M} \times \mathbb{A}^n \dashrightarrow \mathbb{A}^q$ defined by the final results of β. In this sense, the “computational problem” solved by β is given by the final results of β.

Being robust becomes now an architectural requirement for the circuit β and for its output. Robustness implies *well behavedness under restrictions* in the following sense:

Let $\mathcal{N}$ be a constructible subset of $\mathcal{M}$ and suppose that (β, Ω) is robust. Then Proposition 4 implies that the restriction $\Omega|_{\mathcal{N} \times \mathbb{A}^n}$ of the constructible map Ω to $\mathcal{N} \times \mathbb{A}^n$ is still a geometrically robust constructible map.

This implies that (β, Ω) induces a by $\mathcal{N}$ parameterized arithmetical circuit $\beta_{\mathcal{N}}$. Observe that $\beta_{\mathcal{N}}$ may become inconsistent. If $\beta_{\mathcal{N}}$ is consistent then $(\beta_{\mathcal{N}}, \Omega|_{\mathcal{N} \times \mathbb{A}^n})$ is robust. The nodes where the evaluation of $\beta_{\mathcal{N}}$ fails correspond to divisions of zero by zero which may be replaced by so called approximative algorithms having unique limits (see Section 3.3.2). These limits are given by the map $\Omega|_{\mathcal{N} \times \mathbb{A}^n}$. We call $(\beta_{\mathcal{N}}, \Omega|_{\mathcal{N} \times \mathbb{A}^n})$, or simply $\beta_{\mathcal{N}}$, the *restriction* of (β, Ω) or β to $\mathcal{N}$.

We say that the parameterized arithmetic circuit β is *totally division–free* if any division node of β corresponds to a division by a non–zero complex scalar.

We call β *essentially division–free* if only parameter nodes are labelled by divisions. Thus the property of β being totally division–free implies that β is essentially division–free, but not vice versa. Moreover, if β is totally division-free, the rational map given by the intermediate results of β is polynomial and therefore a geometrically robust constructible map. Thus, any by $\mathcal{M}$ parameterized, totally division–free circuit is in a natural way robust.

In the sequel, we shall need the following elementary fact.

Lemma 7 *Let notations and assumptions be as before and suppose that the parameterized arithmetic circuit β is robust. Then all intermediate results of β are polynomials in $X_1, \ldots, X_n$ over the $\mathbb{C}$–algebra of geometrically robust constructible functions defined on $\mathcal{M}$.*

Proof. Without loss of generality we may assume that $\mathcal{M}$ is irreducible. Let ρ be a node of β which computes the intermediate result $G_\rho : \mathcal{M} \times \mathbb{A}^n \to \mathbb{A}^1$. Definition 6 (i) and the irreducibility of $\mathcal{M}$ imply that G_ρ is a polynomial of $\mathbb{C}(\overline{\mathcal{M}})[X_1, \ldots, X_n]$. Observe that any $x \in \mathbb{A}^n$ induces a geometrically robust constructible map $\mathcal{M} \to \mathbb{A}^1$ whose value at the point $u \in \mathcal{M}$ is $G_\rho(u, x)$. Using interpolation at suitable points of $\mathbb{A}^n$, we see that the coefficients of the polynomial G_ρ are geometrically robust constructible functions with domain of definition $\mathcal{M}$. ■

The statement of this lemma should not lead to confusions with the notion of an essentially division–free parameterized circuit. We say just that the intermediate results of β are polynomials in $X_1, \ldots, X_n$ and do not restrict the type of arithmetic operations contained in β.

Whether a division of a polynomial by one of its factors may always be substituted efficiently by additions and multiplications is an important issue in Theoretical Computer Science (compare [Str73]).

To our parameterized arithmetic circuit β we may associate different complexity measures and models. In this paper we shall mainly be concerned with *sequential computing time*, measured by the *size* of β. Here we refer with “size” to the number of internal nodes of β which count for the given complexity measure. Our basic complexity measure is the *non–scalar* one (also called *Ostrowski measure*) over the ground field $\mathbb{C}$. This means that we count, at unit costs, only essential multiplications and divisions (involving basic parameters or input variables in both arguments in the case of a multiplication and in the second argument in the case of a division), whereas $\mathbb{C}$–linear operations are free (see [BCS97] for details).

Let γ_1 and γ_2 be two robust parameterized arithmetic circuits with parameter domain $\mathcal{M}$ and suppose that there is given a one–to–one correspondence λ which identifies the output nodes of γ_1 with the input nodes of γ_2 (thus they must have the same number). Using this identification we may now join the circuit γ_1 with the circuit γ_2 in order to obtain a new parameterized arithmetic circuit $\gamma_2 *_\lambda \gamma_1$ with parameter domain $\mathcal{M}$. The circuit $\gamma_2 *_\lambda \gamma_1$ has the same input nodes as γ_1 and the same output nodes as γ_2 and one deduces easily from Lemma 7 and Proposition 4 that the circuit $\gamma_2 *_\lambda \gamma_1$ is robust and represents a composition of the rational maps defined by γ_1 and γ_2, if $\gamma_2 *_\lambda \gamma_1$ is consistent. The (consistent) circuit $\gamma_2 *_\lambda \gamma_1$ is called the (consistent) *join* of γ_1 with γ_2.

Observe that the final results of a given robust parameterized arithmetic circuit may constitute a vector of parameters. The join of such a circuit with another robust parameterized arithmetic circuit, say β, is again a robust parameterized arithmetic circuit which is called an *evaluation* of β. Hence, mutatis mutandis, the notion of join of two routines includes also the case of circuit evaluation.

We describe now how, based on its semantics, a given parameterized arithmetic circuit β with parameter domain $\mathcal{M}$ may be rewritten as a new circuit over $\mathcal{M}$ which computes the same final results as β.

The resulting two rewriting procedures, called *reduction* and *broadcasting*, will neither be unique, nor generally confluent. To help understanding, the reader may suppose that there is given an (efficient) algorithm which allows identity checking between intermediate results of β. However, we shall not make explicit reference to this assumption. We are now going to explain the first rewriting procedure.

Suppose that the parameterized arithmetic circuit β computes at two different nodes, say ρ and ρ', the same intermediate result. Assume first that ρ neither depends on ρ', nor ρ' on ρ. Then we may erase ρ' and its two ingoing edges (if ρ' is an internal node) and draw an outgoing edge from ρ to any other node of β which is reached by an outgoing edge of ρ'. If ρ' is an output node, we label ρ also as output node. Observe that in this manner a possible indexing of the output nodes of β may become changed but not the final results of β themselves.

Suppose now that ρ' depends on ρ. Since the DAG β is acyclic, ρ does not depend on ρ'. We may now proceed in the same way as before, erasing the node ρ'.

Let β' be the parameterized arithmetic circuit obtained, as described before, by erasing the node ρ'. Then we call β' a *reduction* of β and call the way we obtained β' from β a *reduction step*. A *reduction procedure* is a sequence of successive reduction steps.

One sees now easily that a reduction procedure applied to β produces a new parameterized arithmetic circuit β^* (also called a *reduction* of β) with the same basic parameter and input nodes, which computes the same final results as β (although their possible indexing may be changed). Moreover, if β is a robust parameterized circuit, then β^* is robust too. Observe also that in the case of robust parameterized circuits our reduction commutes with restriction.

Now we introduce the second rewriting procedure.

Let assumptions and notations be as before and let be given a set P of nodes of β and a robust parameterized arithmetic circuit γ with parameter domain $\mathcal{M}$ and $\#P$ input nodes, namely for each $\rho \in P$ one which becomes labelled by a new input variable Y_ρ. We obtain a new parameterized arithmetic circuit, denoted by $\gamma *_P \beta$, when we join γ with β, replacing for each $\rho \in P$ the input node of γ, which is labelled by the variable Y_ρ, by the node ρ of β. The output nodes of β constitute also the output nodes of $\gamma *_P \beta$. Thus β and $\gamma *_P \beta$ compute the same final results. Observe that $\gamma *_P \beta$ is robust if it is consistent. We call the circuit $\gamma *_P \beta$ and all its reductions *broadcastings* of β. Thus broadcasting a robust parameterized arithmetic circuit means rewriting it using only valid polynomial identities.

If we consider arithmetic circuits as computer programs, then reduction and broadcasting represent a kind of program transformations.

3.1.1 A specification language for circuits

Computer programs (or "programmable algorithms") written in high level languages are not the same thing as just "algorithms" in Complexity Theory. Whereas in the uniform view algorithms become implemented by suitable machine models and in the non–uniform view by devices like circuits; specifications and correctness proofs are not treated by the general theory, but only, if necessary, outside of it in a case–by–case ad–hoc manner. The meaning of "algorithm" in Complexity Theory is therefore of syntactic nature.

On the other hand, computer programs, as well as their subroutines (modules) include specifications and correctness proofs, typically written in languages organized by a hierarchy of different abstraction levels. In this sense *programmable algorithms* become equipped with semantics. This is probably the main difference between Complexity Theory and Software Engineering.

In this paper, we are only interested in algorithms which in some sense are programmable. The routines of our computation model will operate on parameterized arithmetic circuits (see Section 3.3). Therefore we are now going to fix a (many–sorted) first–order specification language $\mathcal{L}$ for these circuits.

The language $\mathcal{L}$ will include the following non–logical symbols:

- $0, 1, +, -, \times$, and a constant for each complex number,
- variables

$$n_1, \ldots, n_s \ldots$$

$$\alpha^{(1)}, \ldots, \alpha^{(t)} \ldots$$

$$\beta_1, \ldots, \beta_k \ldots$$

$$\rho_1, \ldots, \rho_l \ldots$$

$$\mathcal{M}_1, \ldots, \mathcal{M}_k \ldots$$

$$U^{(1)}, \ldots, U^{(m)} \ldots$$

$$X^{(1)}, \ldots, X^{(h)} \ldots$$

$$Y^{(1)}, \ldots, Y^{(q)} \ldots$$

to denote non–negative integers and vectors of them, robust parameterized arithmetic circuits, their nodes, their parameter domains, their parameter instances, their input variable vectors and instances of input variable vectors in suitable affine spaces,

- suitable binary predicate symbols to express relations like "ρ is a node of the circuit β", "multiplication is the label of the node ρ of the circuit β", "$\mathcal{M}$ is the parameter domain of the circuit β", "U is a parameter instance of the circuit β", "r is a non–negative integer and the vector length of U is r", "X is the input variable vector of the circuit β" and "n is a non–negative integer and the vector length of X is n",
- a ternary predicate symbol to express "ρ_1 and ρ_2 are nodes of the circuit β and there is an edge of β from ρ_1 to ρ_2",
- binary function symbols to express "U is a parameter instance, k is a natural number and U_k is the k–th entry of U" and "X is an input variable vector, n is a natural number and X_n is the n–th entry of X" and "Y is a variable vector instance, n is a natural number and Y_n is the n–th entry of Y",
- a unary function and a binary predicate symbol to express "the set of output nodes of the circuit β" and "ρ is an output node of the circuit β"
- a quaternary function symbol $G_\rho(\beta; U; X)$ to express "ρ is a node of the circuit β, U is a parameter instance and X is the input variable vector of β and $G_\rho(\beta; U; X)$ is the intermediate result of β at the node ρ and the parameter instance U",
- a predicate symbol for equality for any of the sorts just introduced.

For the treatment of non–negative integers we add the Presburger arithmetic to our first–order specification language $\mathcal{L}$.

At our convenience we may add new function and predicate symbols and variable sorts to $\mathcal{L}$. Typical examples are for β a circuit, U a parameter instance, X the input variable vector and $\rho, \rho_1, \ldots, \rho_m$ nodes of β:
"degree of $G_\rho(\beta; U; X)$" and "the vector lengths of X and Y are equal (say n) and Y is a point of the closed subvariety of $\mathbb{A}^n$ defined by the polynomials $G_{\rho_1}(\beta; U; X)$, $\ldots$, $G_{\rho_m}(\beta; U; X)$".

In the same spirit, we may increase the expressive power of $\mathcal{L}$ in order to be able to express for a robust parameterized circuit β with irreducible parameter domain, U a parameter instance, X the input variable vector, ρ a node of β and α a vector of non–negative integers of the same length as X (say n), "the coefficient of the monomial X^α occurring in the polynomial $G_\rho(\beta; U; X)$" (recall Lemma 7). Here we denote for $X := (X_1, \ldots, X_n)$ and $\alpha := (\alpha_1, \ldots, \alpha_n)$ by X^α the monomial $X^\alpha := X_1^{\alpha_1}, \ldots, X_n^{\alpha_n}$.

The semantics of the specification language $\mathcal{L}$ is determined by the universe of all robust parameterized arithmetic circuits, where we interpret all variables, function symbols and predicates as explained before. We call this universe the *standard model* of $\mathcal{L}$. The set of all closed formulas of $\mathcal{L}$ which are true in this model form the *elementary theory* of $\mathcal{L}$.

3.2 Generic computations

In the sequel, we shall use ordinary arithmetic circuits over $\mathbb{C}$ as *generic computations* [BCS97] (also called *computation schemes* in [Hei89]). The indegree zero nodes of these arithmetic circuits are labelled by scalars and parameter and input variables.

The aim is to represent different parameterized arithmetic circuits of similar size and appearance by different specializations (i.e., instantiations) of the parameter variables in one and the same generic computation. For a suitable specialization of the parameter variables, the original parameterized arithmetic circuit may then be recovered by an appropriate reduction process applied to the specialized generic computation.

This alternative view of parameterized arithmetic circuits will be fundamental for the design of routines of the branching–free computation model we are going to describe in Section 3.3.2. The routines of our computation model will operate on robust parameterized arithmetic circuits and their basic ingredients will be subroutines which calculate parameter instances of suitable, by the model previously fixed, generic computations. These generic computations will be organized in finitely many families which will only depend on a constant number of discrete parameters. These discrete families constitute the basic building block of our model for branching–free computation.

We shall now exemplify these abstract considerations in the concrete situation of the given parameterized arithmetic circuit β. Mutatis mutandis we shall follow the exposition of [KP96], Section 2. Let $l, L_0, \ldots, L_{l+1}$ with $L_0 \geq r + n + 1$ and $L_{l+1} \geq q$ be given natural numbers. Without loss of generality we may suppose that the non–scalar depth of β is positive and at most l, and that β has an oblivious levelled structure of $l + 2$ levels of width at most $L_0, \ldots, L_{l+1}$. Let $U_1, \ldots, U_r$ be new indeterminates (they will play the role of a set of "special" parameter variables which will only be instantiated by $\pi_1, \ldots, \pi_r$).

We shall need the following indexed families of "scalar" parameter variables (which will only be instantiated by complex numbers):

- for $n + r < j \leq L_0$ the indeterminate V_j;
- for $1 \leq i \leq l$, $1 \leq j \leq L_i$, $0 \leq h \leq i$, $1 \leq k \leq L_h$, the indeterminates $A_{i,j}^{(h,k)}$, $B_{i,j}^{(h,k)}$ and $S_{i,j}$, $T_{i,j}$;

- for $1 \leq j \leq L_{l+1}$, $1 \leq k \leq L_l$ the indeterminate C_j^k.

We consider now the following function Q which assigns to every pair (i,j), $1 \leq i \leq l$, $1 \leq j \leq L_i$ and $(l+1,j)$, $1 \leq j \leq L_{l+1}$ the rational expressions defined below:

$$Q_{0,1} := U_1, \ldots, Q_{0,r} := U_r,$$

$$Q_{0,r+1} := X_1, \ldots, Q_{0,r+n} := X_n,$$

$$Q_{0,r+n+1} := V_{r+n+1}, \ldots, Q_{0,L_0} := V_{L_0}.$$

For $1 \leq i \leq l$ and $1 \leq j \leq L_i$ the value $Q_{i,j}$ of the function Q is recursively defined by

$$Q_{i,j} := S_{i,j}(\sum_{\substack{0\leq h<i \\ 1\leq k\leq L_h}} A_{i,j}^{(h,k)} Q_{h,k} \ \cdot \sum_{\substack{0\leq k'<i \\ 1\leq k'\leq L_{h'}}} B_{i,j}^{(h',k')} Q_{h',k'}) \ +$$

$$T_{i,j}(\sum_{\substack{0\leq h<i \\ 1\leq k\leq L_h}} A_{i,j}^{(h,k)} Q_{h,k} \ / \sum_{\substack{0\leq h'<i \\ 1\leq k'\leq L_{h'}}} B_{i,j}^{(h',k')} Q_{h',k'}).$$

Finally, for $(l+1,j)$, $1 \leq j \leq L_{l+1}$ we define $Q_{(l+1,j)} := \sum_{1\leq k\leq L_l} C_j^k Q_{l,k}$.

We interpret the function Q as a (consistent) ordinary arithmetic circuit, say Γ, over $\mathbb{Z}$ (and hence over $\mathbb{C}$) whose indegree zero nodes are labelled by the "standard" input variables $X_1, \ldots, X_n$, the special parameter variables $U_1, \ldots, U_r$ and the scalar parameter variables just introduced.

We consider first the result of instantiating the scalar parameter variables contained in Γ by complex numbers. We call such an instantiation a *specialization* of Γ. It is determined by a point in a suitable affine space. Not all possible specializations are *consistent*, giving rise to an assignment of a rational function of $\mathbb{C}(U_1, \ldots, U_r, X_1, \ldots, X_n)$ to each node of Γ as intermediate result.

We call the specializations which produce a failing assignment *inconsistent*. If in the context of a given specialization of the scalar parameter variables of Γ we instantiate for each index pair (i,j), $1 \leq i \leq l$, $1 \leq j \leq L_i$ the variables $S_{i,j}$ and $T_{i,j}$ by two different values from $\{0,1\}$, the labelled directed acyclic graph Γ becomes an ordinary arithmetic circuit over $\mathbb{C}$ of non–scalar depth at most l and non–scalar size at most $L_1 + \cdots + L_l$ with the inputs $U_1, \ldots, U_r, X_1, \ldots, X_n$.

We may now find a suitable specialization of the circuit Γ into a new circuit Γ' over $\mathbb{C}$ such that the following condition is satisfied:
the (by $\mathcal{M}$) parameterized circuit obtained from Γ' by replacing the special parameter variables $U_1, \ldots, U_r$ by $\pi_1, \ldots, \pi_r$, is consistent and can be reduced to the circuit β.

We may consider the circuit Γ as a generic computation which allows to recover β by means of a suitable specialization of its scalar and special parameter variables into complex numbers and basic parameters $\pi_1, \ldots, \pi_r$ and by means of circuit reductions. Moreover, any by $\mathcal{M}$ parameterized, consistent arithmetic circuit of non–scalar depth at most l, with inputs $X_1, \dots, X_n$ and q outputs, which has an oblivious level structure with $l+2$ levels of width at most $L_0, \dots, L_{l+1}$, may be recovered from Γ by suitable specializations and reductions (see [BCS97], Chapter 9 for more details on generic computations).

3.3 A model for branching–free computation.

3.3.1 Requirements to be satisfied by our branching–free computation model. Informal discussion.

We are now going to introduce a model of branching–free computation with parameterized arithmetic circuits. We shall first require that the routines of this computation model should be well behaved under restrictions of the inputs. We discuss this issue first informally.

Suppose for the moment that our branching–free computation model is already established. Then its routines transform a given parameterized arithmetic (input) circuit into another parameterized (output) circuit such that both circuits have the same parameter domain. Applied to a given parameterized input circuit, a routine of our computation model generates by means of its intermediate steps a DAG of parameterized arithmetic circuits, one contained in the other, which have all the same parameter domain.

Let $\mathcal{A}$ be a routine of our branching–free computation model and consider the previously introduced parameterized circuit β. Let $\mathcal{N}$ be a constructible subset of $\mathcal{M}$ and suppose that β is an admissible input for the routine $\mathcal{A}$. Then $\mathcal{A}$ produces on input β a parameterized arithmetic output circuit with parameter domain $\mathcal{M}$ which we denote by $\mathcal{A}(\beta)$. In order to formulate for the routine $\mathcal{A}$ the requirement of well behavedness under restriction of the inputs, we must be able to restrict β and $\mathcal{A}(\beta)$ to $\mathcal{N}$. Thus β and $\mathcal{A}(\beta)$ should be *robust*, $\beta_{\mathcal{N}}$ should be a consistent admissible input circuit for $\mathcal{A}$ and $\mathcal{A}(\beta_{\mathcal{N}})$ should be consistent too.

Our first architectural requirement on the routine $\mathcal{A}$ may now be formulated as follows:

> *The parameterized arithmetic circuit $\mathcal{A}(\beta_{\mathcal{N}})$ can be recovered from $\mathcal{A}(\beta)$ by restriction to $\mathcal{N}$ and circuit reduction.*

We call this requirement *well behavedness under restrictions*.

The routine $\mathcal{A}$ performs with the parameterized arithmetic circuit β a transformation whose crucial feature is that only nodes which depend on the inputs $X_1, \dots, X_n$ of β become modified, whereas parameter nodes remain substantially preserved. This needs an explicitation.

Suppose that β has t essential parameter nodes. Then the essential parameters (intermediate results) of β associated with these nodes define a geometrically robust constructible map $\theta : \mathcal{M} \to \mathbb{A}^t$. The image $\mathcal{T}$ of θ is a constructible subset of $\mathbb{A}^t$. We require now that, as far as $\mathcal{A}$ performs arithmetic operations with parameters of β, $\mathcal{A}$ does it only with essential ones, and that all essential parameters of $\mathcal{A}(\beta)$ are obtained in this way. Further we require that there exists a geometrically robust constructible map ν defined on $\mathcal{T}$ (e.g., a polynomial map) such that the results of these arithmetic operations occur as entries of the composition map $\nu \circ \theta$. From Proposition 4 we deduce that $\nu \circ \theta$ is a geometrically robust constructible map.

Our basic construction method of routines will be recursion. A routine of our computation model which can be obtained in this way is called *recursive*.

Suppose now that $\mathcal{A}$ is a recursive routine of our computation model. Then $\mathcal{A}$ should be organized in such a way that for each internal node ρ of β, which depends on at least one input, there exists a set of nodes of $\mathcal{A}(\beta)$, also denoted by ρ, with the following property:
the elements of the set ρ of nodes of $\mathcal{A}(\beta)$ represent the outcome of the action of $\mathcal{A}$ at the node ρ of β.

We fix now a node ρ of β which depends on at least one input. Let G_ρ be the intermediate result associated with the node ρ of β and let F_ρ be a vector whose entries are the intermediate results of $\mathcal{A}(\beta)$ at the nodes contained in the set ρ of nodes of $\mathcal{A}(\beta)$. Thus F_ρ is a vector of rational functions in a suitable tuple of (standard) variables, say X'.

Recall that by assumption β and $\mathcal{A}(\beta)$ are robust parameterized arithmetic circuits with parameter domain $\mathcal{M}$. Therefore we deduce from Lemma 7 that G_ρ and the entries of F_ρ are in fact polynomials in $X_1, \ldots, X_n$ and X', respectively, and that their coefficients are geometrically robust functions defined on $\mathcal{M}$.

As part of our second and main requirement of our computation model we demand now that $\mathcal{A}$ satisfies at the node ρ of β the following isoparametricity condition:

(i) for any two parameter instances u_1 and u_2 of $\mathcal{M}$ the assumption

$$G_\rho(u_1, X_1, \ldots, X_n) = G_\rho(u_2, X_1, \ldots, X_n)$$

implies

$$F_\rho(u_1, X') = F_\rho(u_2, X').$$

Let θ_ρ be the coefficient vector of G_ρ and observe that θ_ρ is a geometrically robust constructible map defined on $\mathcal{M}$, whose image, say $\mathcal{T}_\rho$, is an irreducible constructible subset of a suitable affine space.

Since the first–order theory of the algebraically closed field $\mathbb{C}$ admits quantifier elimination, one concludes easily that condition (i) is satisfied if and only if there exists a constructible map σ_ρ defined on $\mathcal{T}_\rho$ such that the composition map $\sigma_\rho \circ \theta_\rho$ (which is also constructible) represents the coefficient vector of (all entries of) F_ρ.

In the sequel we shall need that the dependence σ_ρ of the coefficient vector of F_ρ on the coefficient vector of G_ρ is in some stronger sense uniform (and not just constructible). Therefore we include the following condition in our requirement:

(ii) the constructible map σ_ρ is geometrically robust.

The map σ_ρ is uniquely determined by condition (i). Moreover, the map σ_ρ depends on the (combinatorial) labelled DAG structure of β below the node ρ, but not directly on the basic parameters $\pi_1, \ldots, \pi_r$. This is the essence of the isoparametric nature of conditions (i) and (ii). We shall therefore require that our recursive routine is *isoparametric* in this sense, i.e., that $\mathcal{A}$ satisfies conditions (i) and (ii) at any internal node ρ of β which depends at least on one input.

Observe that the geometrically robust constructible map σ_ρ (which depends on β as well as on ρ) is not an artifact, but emerges naturally from the recursive construction of a circuit semantic within the paradigm of object–oriented programming. To explain this, let notations and assumptions be as before and suppose that $\mathcal{A}$ is a isoparametric recursive routine of our model and that we apply $\mathcal{A}$ to the robust parameterized arithmetic circuit β. Let ρ again be a node of β which depends at least on one input. Let u be a parameter instance of $\mathcal{M}$ and denote by $\beta^{(u)}$, $G_\rho^{(u)}$, $\mathcal{A}(\beta)^{(u)}$ and $F_\rho^{(u)}$ the instantiations of β, G_ρ, $\mathcal{A}(\beta)$ and F_ρ at u (observe that the intermediate results of $\beta^{(u)}$ and $\mathcal{A}(\beta)^{(u)}$ are well defined although we do not require that these circuits are consistent). Then the intermediate results of $\mathcal{A}(\beta)^{(u)}$ contained in $F_\rho^{(u)}$ depend only on the intermediate result $G_\rho^{(u)}$ of $\beta^{(u)}$ and not on the parameter instance u itself. In this spirit we may consider the sets $\Gamma_\rho := \{G_\rho^{(u)} \ ; \ u \in \mathcal{M}\}$ and $\Phi_\rho := \{F_\rho^{(u)} \ ; \ u \in \mathcal{M}\}$ as abstract data types and β and $\mathcal{A}(\beta)$ as syntactic descriptions of two abstraction functions which associate to any concrete object $u \in \mathcal{M}$ the abstract objects $G_\rho^{(u)}$ and $F_\rho^{(u)}$, respectively. The identity map $id_{\mathcal{M}} : \mathcal{M} \to \mathcal{M}$ induces now an *abstract function* [Mey00] from Γ_ρ to Φ_ρ, namely $\sigma_\rho : \Gamma_\rho \to \Phi_\rho$. In this terminology, $id_{\mathcal{M}}$ is just an implementation of σ_ρ. If we now consider that each recursive step of the routine $\mathcal{A}$ on input β has to be realized by some routine of the object–oriented programming paradigm, we arrive to a situation which requires the existence of a geometrically robust constructible map $\sigma_\rho : \Gamma_\rho \to \Phi_\rho$ as above. If we require additionally that this routine is branching–free, the constructible map σ_ρ must be geometrically robust. By the way, this last requirement is also numerically meaningful.

We may interpret the map $\sigma_\rho : \Gamma_\rho \to \Phi_\rho$ also as an ingredient of a specification of the recursive routine $\mathcal{A}$. The map σ_ρ may be thought as an operational specification which determines F_ρ in function of G_ρ. A weaker specification would be a descriptive one which relates G_ρ and F_ρ without determining F_ρ from G_ρ completely.

In order to motivate the requirement that the recursive routine $\mathcal{A}$ should be isoparametric, we shall consider the following condition for recursive routines which we call *well behavedness under reductions*.

We only outline here this condition and leave the details until Section 3.3.2.

Suppose now that we apply a reduction procedure to the robust parameterized input circuit β producing thus another robust, by $\mathcal{M}$ parameterized circuit β^* which computes the same final results as β. Then the reduced circuit β^* should also be an admissible input for the routine $\mathcal{A}$. We call the recursive routine $\mathcal{A}$ *well behaved under reductions* if on input β it is possible to extend the given reduction procedure to the output circuit $\mathcal{A}(\beta)$ in such a way, that the extended reduction procedure, applied to $\mathcal{A}(\beta)$, reproduces the circuit $\mathcal{A}(\beta^*)$.

Obviously well behavedness under reductions limits the structure of $\mathcal{A}(\beta)$. Later, in Section 3.3.2, we shall see that, cum grano salis, any recursive routine, which is well behaved under restrictions and reductions, is necessarily isoparametric. Since well behavedness under restrictions and reductions are very natural quality attributes for routines which transform robust parameterized arithmetic circuits, the weaker requirement, namely that recursive routines should be isoparametric, turns out to be well motivated.

In Section 3.3.2, we shall formally introduce our branching–free computation model. We postpone for then the precise definition of the notion of well behavedness under reductions.

There exists a second reason to restrict the recursive routines of our branching–free computation model to isoparametric ones. Isoparametric recursive routines have considerable advantages for program specification and verification by means of Hoare Logics (see [Apt81]). We shall come back to this issue in Section 3.3.2.

3.3.2 The branching–free computation model

The computation model we are going to introduce in this and the next subsection will be comprehensive enough to capture the essence of all known circuit based elimination algorithms in effective algebraic geometry and, mutatis mutandis, also of all other (linear algebra and truncated rewriting) elimination procedures (see Sections 3.3.3, 4, [Mor03], [Mor05], and the references cited therein, and for truncated rewriting methods especially [DFGS91]). The only algorithm from symbolic arithmetic circuit manipulation which will escape from our model is the Baur–Strassen gradient computation [BCS97], Chapter 7.2.

In the sequel we shall distinguish sharply between the notions of input variable and parameter and the corresponding categories of circuit nodes.

Input variables, called "standard", will occur in parameterized arithmetic circuits and generic computations. The input variables of generic computations will appear subdivided in three sorts, namely as "parameter", "argument" and "standard" input variables.

The branching–free computation model we are going to introduce in this subsection will assume different *shapes*, each shape being determined by a finite number of a priori given *discrete* (i.e., by tuples of natural numbers indexed) families of generic computations. The labels of the inputs of the ordinary arithmetic circuits which represent these generic computations will become subdivided into *parameter*, *argument* and *standard* input variables. We shall use the letters like $U, U', U'', \dots$ and W, W', W'' to denote vectors of parameters, $Y, Y', Y'', \dots$ and Z, Z', Z'' to denote vectors of argument and $X, X', X'', \dots$ to denote vectors of standard input variables (see Section 3.2).

We shall not write down explicitly the indexations of our generic computations by tuples of natural numbers. Generic computations will simply be distinguished by subscripts and superscripts, if necessary.

Ordinary arithmetic circuits of the form

$$\begin{array}{lll} R_{X_1}(W_1;X^{(1)}), & R_{X_2}(W_2;X^{(2)}), & \dots \\ R'_{X_1}(W_{1'};X^{(1')}), & R'_{X_2}(W_{2'};X^{(2')}), & \dots \\ \dots & \dots & \dots \end{array}$$

represent a first type of a discrete family of generic computations (for each variable $X_1, X_2, \dots, X_n$ we suppose to have at least one generic computation). Other types of families of generic computations are of the form

$$\begin{array}{llll} R_{+}(W;U,Y;X), & R'_{+}(W';U',Y';X'), & R''_{+}(W'';U'',Y'';X'') & \dots \\ R_{\cdot/}(W;U,Y;X), & R'_{\cdot/}(W';U',Y';X'), & R''_{\cdot/}(W'';U'',Y'';X'') & \dots \\ R_{add}(W;Y,Z;X), & R'_{add}(W';Y',Z';X'), & R''_{add}(W'';Y'',Z'';X'') & \dots \\ R_{mult}(W;Y,Z;X), & R'_{mult}(W';Y',Z';X'), & R''_{mult}(W'';Y'',Z'';X'') & \dots \end{array}$$

and

$$R_{div}(W;Y,Z;X), \quad R'_{div}(W';Y',Z';X'), \quad R''_{div}(W'';Y'',Z'';X'') \quad \dots.$$

Here the subscripts refer to addition of, and multiplication or division by a parameter (or scalar) and to essential addition, multiplication and division. A final type of families of generic computations is of the form

$$R(W;Y;X),\ \ R'(W';Y';X'),\ \ R''(W'';Y'';X''),\dots$$

We recall from Section 3.3.1 that the objects handled by the routines of any shape of our computation model will always be robust parameterized arithmetic circuits. The inputs of these circuits will only consist of standard variables.

From now on we have in mind a previously fixed shape when we refer to the branching–free computation model we are going to introduce. We start with a given finite set of discrete families of generic computations which constitute a shape as described before.

A fundamental issue is how we recursively transform a given input circuit into another one with the same parameter domain. During such a transformation we make an iterative use of previously fixed generic computations. On their turn these determine the corresponding *recursive routine* of our branching–free computation model.

We consider again our input circuit β. We suppose that we have already chosen for each node ρ, which depends at least on one of the input variables $X_1, \dots, X_n$, a generic computation

$$R^{(\rho)}_{X_i}(W_\rho;X^{(\rho)}),$$

$$\begin{gathered} R_{+}^{(\rho)}(W_\rho; U_\rho, Y_\rho; X^{(\rho)}), \\ R_{\cdot/}^{(\rho)}(W_\rho; U_\rho, Y_\rho; X^{(\rho)}), \\ R_{add}^{(\rho)}(W_\rho; Y_\rho, Z_\rho; X^{(\rho)}), \\ R_{mult}^{(\rho)}(W_\rho; Y_\rho, Z_\rho; X^{(\rho)}), \\ R_{div}^{(\rho)}(W_\rho; Y_\rho, Z_\rho; X^{(\rho)}), \end{gathered}$$

and that this choice was made according to the label of ρ, namely $X_i, 1 \leq i \leq n$, or addition of, or multiplication or division by an essential parameter, or essential addition, multiplication or division. Here we suppose that U_ρ is a single variable, whereas W_ρ, Y_ρ, Z_ρ and $X^{(\rho)}$ may be arbitrary vectors of variables.

Furthermore, we suppose that we have already precomputed for each node ρ of β, which depends at least on one input, a vector w_ρ of geometrically robust constructible functions defined on $\mathcal{M}$. If ρ is an input node we assume that w_ρ is a vector of complex numbers. Moreover, we assume that the length of w_ρ equals the length of the variable vector W_ρ. We call the entries of w_ρ the *parameters at the node* ρ of the routine $\mathcal{A}$ applied to the input circuit β.

We are now going to develop the routine $\mathcal{A}$ step by step. The routine $\mathcal{A}$ takes over all computations of β which involve only parameter nodes, without modifying them.

Consider an arbitrary internal node ρ of β which depends at least on one input. The node ρ has two ingoing edges which come from two other nodes of β, say ρ_1 and ρ_2. Suppose that the routine $\mathcal{A}$, on input β, has already computed two results, namely F_{ρ_1} and F_{ρ_2}, corresponding to the nodes ρ_1 and ρ_2. Suppose inductively that these results are vectors of polynomials depending on those standard input variables that occur in the vectors of the form $X^{(\rho')}$, where ρ' is any predecessor node of ρ. Furthermore, we assume that the coefficients of these polynomials constitute the entries of a geometrically robust, constructible map defined on $\mathcal{M}$. Finally we suppose that the lengths of the vectors F_{ρ_1} and Y_ρ (or U_ρ) and F_{ρ_2} and Z_ρ coincide.

The parameter vector w_ρ of the routine $\mathcal{A}$ forms a geometrically robust, constructible map defined on $\mathcal{M}$, whose image we denote by $\mathcal{K}_\rho$. Observe that $\mathcal{K}_\rho$ is a constructible subset of the affine space of the same dimension as the length of the vectors w_ρ and W_ρ. Denote by κ_ρ the vector of the restrictions to $\mathcal{K}_\rho$ of the canonical projections of this affine space. We consider $\mathcal{K}_\rho$ as a new parameter domain with basic parameters κ_ρ. For the sake of simplicity we suppose that the node ρ is labelled by a multiplication. Thus the corresponding generic computation has the form

$$R_{\cdot/}^{(\rho)}(W_\rho; U_\rho, Y_\rho; X^{(\rho)}) \tag{1}$$

or

$$R_{mult}^{(\rho)}(W_\rho; Y_\rho, Z_\rho; X^{(\rho)}). \tag{2}$$

Let the specialized generic computations

$$R_{\cdot/}^{(\rho)}(\kappa_\rho, U_\rho, Y_\rho, X^{(\rho)}) \text{ and } R_{mult}^{(\rho)}(\kappa_\rho, Y_\rho, Z_\rho, X^{(\rho)})$$

be the by $\mathcal{K}_\rho$ parameterized arithmetic circuits obtained by substituting in the generic computations (1) and (2) for the vector of parameter variables W_ρ the basic parameters κ_ρ. At the node ρ we shall now make the following requirement on the routine $\mathcal{A}$ applied to the input circuit β:

(A) The by $\mathcal{K}_\rho$ parameterized arithmetic circuit which corresponds to the current case, namely

$$R^{(\rho)}_{\cdot/}(\kappa_\rho; U_\rho, Y_\rho; X^{(\rho)})$$

or

$$R^{(\rho)}_{mult}(\kappa_\rho; Y_\rho, Z_\rho; X^{(\rho)}),$$

should be consistent and robust.

Observe that the requirement (A) is automatically satisfied if all the generic computations of our shape are realized by totally division–free ordinary arithmetic circuits.

Assume now that the routine $\mathcal{A}$ applied to the circuit β satisfies the requirement (A) at the node ρ of β.

Let us first suppose that the node ρ is labelled by a multiplication involving an essential parameter. Recall that in this case we assumed earlier that the length of the vector F_{ρ_1} is one, that F_{ρ_1} is an essential parameter of β and that the vectors F_{ρ_2} and Y_ρ have the same length. Joining now with the generic computation $R^{(\rho)}_{\cdot/}(W_\rho; U_\rho, Y_\rho; X^{(\rho)})$ at W_ρ, U_ρ and Y_ρ the previous computations of w_ρ, F_{ρ_1} and F_{ρ_2}, we obtain a parameterized arithmetic circuit with parameter domain $\mathcal{M}$, whose final results are the entries of a vector which we denote by F_ρ.

Suppose now that the node ρ is labelled by an essential multiplication. Recall again that in this second case we assumed earlier the vectors F_{ρ_1} and Y_ρ and F_{ρ_2} and Z_ρ have the same length. Joining with the generic computation

$$R^{(\rho)}_{mult}(W_\rho; Y_\rho, Z_\rho; X^{(\rho)})$$

at W_ρ, Y_ρ and Z_ρ the previous computations of w_ρ, F_{ρ_1} and F_{ρ_2} we obtain also a parameterized arithmetic circuit with parameter domain $\mathcal{M}$, whose final results are the entries of a vector which we denote again by F_ρ.

One deduces easily from our assumptions on w_ρ, F_{ρ_1} and F_{ρ_2} and from the requirement (A) in combination with Lemma 7 and Proposition 4, that in both cases the resulting parameterized arithmetic circuit is robust if it is consistent. The other possible labellings of the node ρ by arithmetic operations are treated similarly. In particular, in case that ρ is an input node labelled by the variable $X_i, 1 \leq i \leq n$, the requirement (A) implies that the ordinary arithmetic circuit $R^{(\rho)}_{X_i}(w_\rho; X^{(\rho)})$ is consistent and robust and that all its intermediate results are polynomials in $X^{(\rho)}$ over $\mathbb{C}$ (although $R^{(\rho)}_{X_i}(w_\rho; X^{(\rho)})$ may contain divisions).

In view of our comments in Section 3.3.1, we call the recursive routine $\mathcal{A}$ (on input β) *well behaved under restrictions* if the requirement (A) is satisfied at any node ρ

of β which depends at least on one input and if joining the corresponding generic computation with w_ρ, F_{ρ_1} and F_{ρ_2} produces a consistent circuit (observe that this last condition is automatically satisfied when the specialized generic computation of (A) is essentially division–free). If the routine $\mathcal{A}$ is well behaved under restrictions, then $\mathcal{A}$ transforms step by step the input circuit β into another consistent robust arithmetic circuit, namely $\mathcal{A}(\beta)$, with parameter domain $\mathcal{M}$.

As a consequence of the recursive structure of $\mathcal{A}(\beta)$, each node ρ of β generates a subcircuit of $\mathcal{A}(\beta)$ which we call the component of $\mathcal{A}(\beta)$ generated by ρ. The output nodes of each component of $\mathcal{A}(\beta)$ form the hypernodes of a hypergraph $\mathcal{H}_{\mathcal{A}(\beta)}$ whose hyperedges are given by the pathes connecting the nodes of $\mathcal{A}(\beta)$ contained in distinct hypernodes of $\mathcal{H}_{\mathcal{A}(\beta)}$. The hypergraph $\mathcal{H}_{\mathcal{A}(\beta)}$ may be shrunk to the DAG structure of β and therefore we denote the hypernodes of $\mathcal{H}_{\mathcal{A}(\beta)}$ in the same way as the nodes of β. Notice that well behavedness under restrictions is in fact a property which concerns the hypergraph $\mathcal{H}_{\mathcal{A}(\beta)}$.

We call $\mathcal{A}$ a (recursive) *parameter routine* if $\mathcal{A}$ does not introduce new standard variables. In the previous recursive construction of the routine $\mathcal{A}$, the parameters at the nodes of β, used for the realization of the circuit $\mathcal{A}(\beta)$, are supposed to be generated by recursive parameter routines.

We are now going to consider another requirement of our recursive routine $\mathcal{A}$, which will lead us to the notion of *isoparametricity* of $\mathcal{A}$.

Let us turn back to the previous situation at the node ρ of the input circuit β. Notations and assumptions will be the same as before. From Lemma 7 we deduce that the intermediate result of β associated with the node ρ, say G_ρ, is a polynomial in $X_1, \dots, X_n$ whose coefficients form the entries of a geometrically robust, constructible map defined on $\mathcal{M}$, say θ_ρ. Let $\mathcal{T}_\rho$ be the image of this map and observe that $\mathcal{T}_\rho$ is a constructible subset of a suitable affine space. The intermediate results of the circuit $\mathcal{A}(\beta)$ at the elements of the hypernode ρ of $\mathcal{H}_{\mathcal{A}(\beta)}$ constitute a polynomial vector which we denote by F_ρ.

We shall now make another requirement at the node ρ on the routine $\mathcal{A}$ applied to the input circuit β:

(B) There exists a geometrically robust, constructible map σ_ρ defined on $\mathcal{T}_\rho$ such that $\sigma_\rho \circ \theta_\rho$ constitutes the coefficient vector of F_ρ.

In view of the comments made in Section 3.3.1 we call the recursive routine $\mathcal{A}$ *isoparametric* (on input β) if requirements (A) and (B) are satisfied at any node ρ of β which depends at least on one input.

Let assumptions and notations be as before and consider again the node ρ of the circuit β. Assume that the recursive routine $\mathcal{A}$ is well behaved under restrictions and denote by τ_ρ the coefficient vector of F_ρ. Observe that τ_ρ is a geometrically robust constructible map defined on $\mathcal{M}$. Assume, furthermore, that $\mathcal{A}$, applied to the circuit β, fulfills the requirement (B) at ρ. Then the topological robustness (which is a consequence of the geometrical robustness) of σ_ρ implies that the following condition is satisfied:

(B') *Let $(u_k)_{k\in\mathbb{N}}$ be a (not necessarily convergent) sequence of parameter instances $u_k \in \mathcal{M}$ and let $u \in \mathcal{M}$ such that $(\theta_\rho(u_k))_{k\in\mathbb{N}}$ converges to $\theta_\rho(u)$. Then the sequence $(\tau_\rho(u_k))_{k\in\mathbb{N}}$ is bounded.*

Suppose now that the recursive routine $\mathcal{A}$ is well behaved under restrictions and satisfies instead of (B) only condition (B') at the node ρ of β. Let $u \in \mathcal{M}$ be an arbitrary parameter instance. Then Theorem 3 implies that τ_ρ takes on the set $\{u' \in \mathcal{M}; \theta_\rho(u') = \theta_\rho(u)\}$ only finitely many values. In particular, for $\mathfrak{M}_u$ being the vanishing ideal of the $\mathbb{C}$–algebra $\mathbb{C}[\theta_\rho]$ at $\theta_\rho(u)$, the entries of τ_ρ are integral over the local $\mathbb{C}$–algebra $\mathbb{C}[\theta_\rho]_{\mathfrak{M}_u}$ (the argument for that relies on Zariski's Main Theorem and is exhibited in [CGH$^+$03], Sections 3.2 and 5.1). This algebraic characterization implies that for given $u \in \mathcal{M}$ all the sequences $(\tau_\rho(u_k))_{k\in\mathbb{N}}$ of condition (B') have only finitely many distinct accumulation points. This shows that requirement (B) and condition (B') are closely related.

Adopting the terminology of [GHMS11] we call $\mathcal{A}$ *coalescent* (on input β), if $\mathcal{A}$ is well behaved under restrictions and satisfies condition (B') for any node ρ of β. Thus isoparametricity implies coalescence for $\mathcal{A}$, but not vice versa. Nevertheless the notions of isoparametricity and coalescence become quite close for recursive routines which are well behaved under restrictions.

Suppose again that the recursive routine $\mathcal{A}$ is well behaved under restrictions. We call $\mathcal{A}$ *well behaved under reductions* (on input β) if $\mathcal{A}(\beta)$ satisfies the following requirement:

> *Let ρ and ρ' be distinct nodes of β which compute the same intermediate results. Then the intermediate results at the hypernodes ρ and ρ' of $\mathcal{H}_{\mathcal{A}(\beta)}$ are identical. Mutatis mutandis the same is true for the computation of the parameters of $\mathcal{A}$ at any node of β.*

Assume that the routine $\mathcal{A}$ is recursive and well behaved under reductions. One verifies then easily that, taking into account the hypergraph structure $\mathcal{H}_{\mathcal{A}(\beta)}$ of $\mathcal{A}(\beta)$, any reduction procedure on β may canonically be extended to a reduction procedure of $\mathcal{A}(\beta)$.

In Section 3.3.1 we claimed that, cum grano salis, the requirement of well behavedness under reductions implies the requirement of isoparametricity for recursive routines. We are going now to prove this.

Let notations and assumptions be as before and let us analyze what happens to the recursive routine $\mathcal{A}$ at the node ρ of β. For this purpose we shall use the following broadcasting argument.

Recall that G_ρ and the entries of F_ρ are the intermediate results of β and $\mathcal{A}(\beta)$ associated with ρ, where ρ is interpreted as a node of the input circuit β in the first case and as a hypernode of $\mathcal{H}_{\mathcal{A}(\beta)}$ in the second one. Moreover recall that

G_ρ is a polynomial in $X_1, \ldots, X_n$, that the geometrically robust, constructible map θ_ρ, defined on $\mathcal{M}$, represents the coefficient vector of G_ρ and that the irreducible constructible set $\mathcal{T}_\rho$ is the image of θ_ρ. Observe that the entries of θ_ρ may be computed from $\pi_1, \ldots, \pi_r$ by a robust arithmetic circuit (e.g., by interpolation of G_ρ in sufficiently generic points of $\mathbb{A}^n$). We consider now the robust parameterized arithmetic circuit γ_ρ which realizes the following trivial evaluation of the polynomial G_ρ:

- compute simultaneously from $\pi_1, \ldots, \pi_r$ all entries of θ_ρ and from $X_1, \ldots, X_n$ all monomials occurring in G_ρ
- compute G_ρ as a linear combination of the monomials of G_ρ using as coefficients the entries of θ_ρ.

The circuit γ_ρ has a single output node, say ρ', which computes the polynomial G_ρ.

Now we paste, as disjointly as possible, the circuit γ_ρ to the circuit β obtaining thus a new robust, parameterized arithmetic circuit β_ρ with parameter domain $\mathcal{M}$. Observe that β_ρ contains β and γ_ρ as subcircuits and that ρ and ρ' are distinct nodes of β_ρ which compute the same intermediate result, namely G_ρ. The entries of θ_ρ are essential parameters of γ_ρ and hence also of β_ρ. We suppose now that β_ρ is, like β, an admissible input for the recursive routine $\mathcal{A}$. Let $F_{\rho'}$ be a vector whose entries are the intermediate results at the nodes of $\mathcal{A}(\beta_\rho)$ contained in the hypernode ρ' of $\mathcal{H}_{\mathcal{A}(\beta_\rho)}$. Analyzing now how $\mathcal{A}$ operates on the structure of the subcircuit γ_ρ of β_ρ, we see immediately that there exists a geometrically robust constructible map σ_ρ defined on $\mathcal{T}_\rho$ such that the composition map $\sigma_\rho \circ \theta_\rho$ constitutes the coefficient vector of $F_{\rho'}$. Since by assumption the recursive routine $\mathcal{A}$ is well behaved under reductions and the intermediate results of β_ρ at the nodes ρ and ρ' consist of the same polynomial G_ρ, we conclude that the intermediate results at the hypernodes ρ and ρ' of $\mathcal{H}_{\mathcal{A}(\beta_\rho)}$ are also the same. Therefore we may assume without loss of generality $F_\rho = F_{\rho'}$. Hence the geometrically robust, constructible map $\sigma_\rho \circ \theta_\rho$ constitutes the coefficient vector of F_ρ.

This proves that the recursive routine $\mathcal{A}$ satisfies, on input β and at the node ρ, the requirement (B). Since β was an arbitrary admissible input circuit for the recursive routine $\mathcal{A}$ and ρ was an arbitrary node of β which depends on at least one input, we may conclude that $\mathcal{A}$ is isoparametric. The only assumption we made to draw this conclusion was that the extended circuit β_ρ is an admissible input for the routine $\mathcal{A}$. This conclusion is however not very restrictive because β and β_ρ compute the same final results.

In Section 3.3.1, we mentioned that isoparametric routines are advantageous for program specification and verification. We are now going to explain this.

Let notations and assumptions be as before and let in particular $\mathcal{A}$ be a recursive routine of our computation model which behaves well under restrictions. Assume that β is an admissible input for $\mathcal{A}$ and consider the specification language $\mathcal{L}$ introduced in Section 3.1.1. Suppose that the routine $\mathcal{A}$ is given by an asserted program Π formulated in the elementary Hoare Logics of $\mathcal{L}$ ([Apt81]). The standard model of the elementary theory of $\mathcal{L}$ provides us with the states which define the semantics of Π. The asserted program Π represents the routine $\mathcal{A}$ as a loop which transforms node by node the labelled DAG structure of β into the labelled DAG structure of $\mathcal{A}(\beta)$.

At each step of the loop a purely syntactic action, namely a graph manipulation, takes place. This action consists of the join of two or more labelled directed acyclic graphs. Simultaneously, in order to guarantee the correctness of the program Π, a loop invariant, formulated in our specification language $\mathcal{L}$, has to be satisfied.

This involves the semantics of $\mathcal{L}$ consisting of the universe of all robust parameterized arithmetic circuits. A loop invariant as above is given by a formula $\bigwedge(\beta_1, \beta_2, \mathcal{M}_1, \rho_1)$ of $\mathcal{L}$ containing the free variables β_1, β_2 for circuits over the same parameter domain $\mathcal{M}_1$ and ρ_1 for a node of β_1 and a linked hypernode of β_2, such that these free variables become instantiated by β, $\mathcal{A}(\beta)$, $\mathcal{M}$ and the node ρ of β or the hypernode ρ of $\mathcal{A}(\beta)$. The variables $U^{(1)}, \ldots, U^{(m)}, \ldots$ and the standard input variable vectors $X^{(1)}, \ldots, X^{(h)}, \ldots$ occur only bounded in $\bigwedge(\beta_1, \beta_2, \mathcal{M}_1, \rho_1)$ and the variables $\rho_1, \ldots, \rho_l, \ldots$ occur all bounded except one, namely ρ_1.

For $\pi := (\pi_1, \ldots, \pi_r)$ and given variables X, X' and ρ expressing a parameter instantiation, the input variable vectors of β and $\mathcal{A}(\beta)$ and a node of β, we denote by $G_\rho(\beta; \pi; X)$ and $F_\rho(\mathcal{A}(\beta); \pi; X')$ the function symbols (or vectors of them) which express the intermediate results of β or $\mathcal{A}(\beta)$ corresponding to ρ.

We require now that any formula of $\mathcal{L}$ built up by $G_{\rho_1}, \ldots, G_{\rho_l}$ and $F_{\rho'_1}, \ldots, F_{\rho'_{l'}}$, and containing only β, $\mathcal{M}$ and ρ_1 as free variables is equivalent to a formula built up only by $G_{\rho_1}, \ldots, G_{\rho_l}$ and $G_{\rho'_1}, \ldots, G_{\rho'_{l'}}$. This implies that in $\mathcal{L}$ the intermediate result F_ρ of $\mathcal{A}(\beta)$ is definable in terms of the intermediate result G_ρ of β. Applied to the node ρ of the concrete circuit β with parameter domain $\mathcal{M}$, this means that for θ_ρ and τ_ρ being the coefficient vectors of $G_\rho(\beta, \pi, X)$ and $F_\rho(\mathcal{A}(\beta), \pi, X')$ and $\mathcal{T}_\rho$ being the image of θ_ρ, there exists a constructible map σ_ρ with domain of definition $\mathcal{T}_\rho$ such that $\tau_\rho = \sigma_\rho \circ \theta_\rho$ holds. In particular, for $u', u'' \in \mathcal{M}$ the assumption $\theta_\rho(u') = \theta_\rho(u'')$ implies $\tau_\rho(u') = \tau_\rho(u'')$.

For the modelling of elimination algorithms this is a reasonable requirement (see Section 4). If we require additionally that the transformation of $G_\rho(\beta, \pi, X)$ into $F_\rho(\mathcal{A}(\beta), \pi, X')$ is branching–free, then the constructible map σ_ρ has to be geometrically robust (see Section 3.3.1).

In terms of the specification language $\mathcal{L}$, this reasoning may be formulated as follows.

Let $\beta_1, \beta_2, \mathcal{M}_1$ and ρ_1 be variables for robust parameterized arithmetic circuits, their parameter domains and their (hyper)nodes. We assume that there exist a formula

$$\Omega(\beta_1, \beta_2, \mathcal{M}_1, \rho_1)$$

in the free variables $\beta_1, \beta_2, \mathcal{M}_1, \rho_1$ such that for any concrete, for $\mathcal{A}$ admissible circuit

β with parameter domain $\mathcal{M}$ and basic parameter vector π and for any node ρ of β the following condition is satisfied:

$(*)$ $\Omega(\beta, \mathcal{A}(\beta), \mathcal{M}, \rho)$ determines the polynomial $F_\rho(\mathcal{A}(\beta), \pi, X')$ in terms of $G_\rho(\beta, \pi, X)$.

If $\mathcal{L}$ and $\mathcal{A}$ satisfy this assumption we say in the spirit of Hoare Logics that $\mathcal{L}$ is *expressive* for the routine $\mathcal{A}$.

Observe that condition $(*)$ guarantees that a postcondition for the circuit $\mathcal{A}(\beta)$ can always be translated into an equivalent precondition for the circuit β.

Let $\mathcal{A}$ and $\mathcal{B}$ be recursive routines as before and suppose that they are well behaved under restrictions and isoparametric or even well behaved under reductions. Assume that $\mathcal{A}(\beta)$ is an admissible input for $\mathcal{B}$. We define the composed routine $\mathcal{B} \circ \mathcal{A}$ in such a way that $(\mathcal{B} \circ \mathcal{A})(\beta)$ becomes the parameterized arithmetic circuit $\mathcal{B}(\mathcal{A}(\beta))$. Since the routines $\mathcal{A}$ and $\mathcal{B}$ are well behaved under restrictions, we see easily that $(\mathcal{B} \circ \mathcal{A})(\beta)$ is a consistent, robust parameterized arithmetic circuit with parameter domain $\mathcal{M}$. From Lemma 7 and Proposition 4 we deduce that $\mathcal{B} \circ \mathcal{A}$ is a isoparametric recursive routine if $\mathcal{A}$ and $\mathcal{B}$ are isoparametric. In case that $\mathcal{A}$ and $\mathcal{B}$ are well behaved under reductions, one verifies immediately that $\mathcal{B} \circ \mathcal{A}$ is also well behaved under reductions. Therefore, under these assumptions, we shall consider $\mathcal{B} \circ \mathcal{A}$ also as a routine of our computation model.

Unfortunately, the composition of two arbitrary coalescent recursive routines need not to be coalescent. Therefore we shall focus in the sequel our attention on isoparametric recursive routines as basic building blocks of the branching–free computation model we are going to introduce.

The identity routine is trivially well behaved under restrictions and reductions and in particular isoparametric.

Let $\mathcal{A}$ and $\mathcal{B}$ be two routines of our computation model and suppose for the sake of simplicity that they are recursive and well behaved under restrictions. Assume that the robust parameterized arithmetic circuit β is an admissible input for $\mathcal{A}$ and $\mathcal{B}$ and that there is given a one–to–one correspondence λ which identifies the output nodes of $\mathcal{A}(\beta)$ with the input nodes of $\mathcal{B}(\beta)$. Often, for a given input circuit β, the correspondence λ is clear by the context. If we limit ourselves to input circuits β where this occurs, we obtain from $\mathcal{A}$ and $\mathcal{B}$ a new routine, called their *join*, which transforms the input circuit β into the output circuit $\mathcal{B}(\beta) *_\lambda \mathcal{A}(\beta)$ (here we suppose that $\mathcal{B}(\beta) *_\lambda \mathcal{A}(\beta)$ is consistent). Analyzing now $\mathcal{B}(\beta) *_\lambda \mathcal{A}(\beta)$, we see that the join of $\mathcal{A}$ with $\mathcal{B}$ is well behaved under restrictions in the most obvious sense. Since by assumption the routines $\mathcal{A}$ and $\mathcal{B}$ are recursive, the circuits $\mathcal{A}(\beta)$ and $\mathcal{B}(\beta)$ inherit from β a superstructure given by the hypergraphs $\mathcal{H}_{\mathcal{A}(\beta)}$ and $\mathcal{H}_{\mathcal{B}(\beta)}$. Analyzing again this situation, we see that any reduction procedure on β can be extended in a canonical way to the circuit $\mathcal{B}(\beta) *_\lambda \mathcal{A}(\beta)$. This means that the join of $\mathcal{A}$ with $\mathcal{B}$ is also well behaved under reductions if the same is true for $\mathcal{A}$ and $\mathcal{B}$. More caution is at order with the notions of isoparametricity and coalescence. In a strict sense, the join of two isoparametric or coalescent recursive routines $\mathcal{A}$ and $\mathcal{B}$ is not necessarily isoparametric or coalescent. However the conditions (B) or (B') are still satisfied between the output nodes of β and $\mathcal{B}(\beta) *_\lambda \mathcal{A}(\beta)$. A routine with one of these two properties is called *output isoparametric* or *output coalescent*, respectively.

The *union* of the routines $\mathcal{A}$ and $\mathcal{B}$ assigns to the input circuit β the juxtaposition of $\mathcal{A}(\beta)$ and $\mathcal{B}(\beta)$. Thus, on input β, the final results of the union of $\mathcal{A}$ and $\mathcal{B}$ are the final results of $\mathcal{A}(\beta)$ and $\mathcal{B}(\beta)$ (taken separately in case of ambiguity). The union of $\mathcal{A}$ and $\mathcal{B}$ behaves well under restrictions and reductions and is isoparametric if the same is true for $\mathcal{A}$ and $\mathcal{B}$.

Observe also that for a recursive routine $\mathcal{A}$ which behaves well under restrictions and reductions the following holds: let β be a robust parameterized arithmetic circuit that broadcasts to a circuit β^* and assume that β and β^* are admissible circuits for $\mathcal{A}$. Then $\mathcal{A}(\beta)$ broadcasts to $\mathcal{A}(\beta^*)$.

From these considerations we conclude that routines, constructed as before by iterated applications of the operations isoparametric recursion, composition, join and union, are still, in a suitable sense, well behaved under restrictions and output isoparametric. If only recursive routines become involved that behave well under reductions, we may also allow broadcastings at the interface of two such operations.

This remains true when we introduce, as we shall do now, in our computational model the following additional type of routine construction.

Let β be the robust, parameterized circuit considered before, and let $R(W;Y;X)$ be a generic computation belonging to our shape list. Let w_β be a precomputed vector of geometrically robust constructible functions with domain of definition $\mathcal{M}$ and suppose that w_β and W have the same vector length and that the entries of w_β are the final results of an output isoparametric parameter routine applied to the circuit β. Moreover suppose that the final results of β form a vector of the same length as Y.

Let $\mathcal{K}$ be the image of w_β. Observe that $\mathcal{K}$ is a constructible subset of the affine space which has the same dimension as the vector length of W. Denote by κ the vector of the restrictions to $\mathcal{K}$ of the canonical projections of this affine space. We denote by $R(\kappa;Y;X)$ the ordinary arithmetic circuit over $\mathbb{C}$ obtained by substituting in the generic computation $R(W;Y;X)$ the vector of parameter variables W by κ. We shall now make the following requirement:

(C) The ordinary arithmetic circuit $R(\kappa;Y;X)$ should be consistent and robust.

Observe that requirement (C) is obsolete when $R(W;Y;X)$ is a totally division–free ordinary arithmetic circuit.

Suppose now that requirement (C) is satisfied. A new routine, say $\mathcal{B}$, is obtained in the following way: on input β the routine $\mathcal{B}$ joins with the generic computation $R(W;Y;X)$ at W and Y the previous computation of w_β and the circuit β.

From Lemma 7 and Proposition 4 we deduce that the resulting parameterized arithmetic circuit $\mathcal{B}(\beta)$ has parameter domain $\mathcal{M}$ and is robust if it is consistent. We shall therefore require that $\mathcal{B}(\beta)$ is consistent (this condition is automatically satisfied if $R(\kappa; Y; X)$ is essentially division–free). One sees immediately that the routine $\mathcal{B}$ is well behaved under restrictions and reductions and is output isoparametric.

From now on we shall always suppose that all our recursive routines are isoparametric and well behaved under restrictions and that requirement (C) is satisfied when we apply this last type of routine construction.

An *elementary routine* of our simplified *branching–free computation model* is finally obtained by the iterated application of all these construction patterns, in particular the last one, isoparametric recursion, composition, join and union. As far as only recursion becomes involved that is well behaved under reductions, we allow also broadcastings at the interface of two constructions. Of course, the identity routine belongs also to our model. The set of all these routines is therefore closed under these constructions and operations.

We call an elementary routine *essentially division–free* if it admits as input only essentially division–free, robust parameterized arithmetic circuits and all specialized generic computations used to compose it are essentially division–free. The outputs of essentially division–free elementary routines are always essentially division–free circuits. The set of all essentially division–free elementary routines is also closed under the mentioned constructions and operations.

We have seen that elementary routines are, in a suitable sense, well behaved under restrictions. In the following statement we formulate explicitly the property of an elementary routine to be output isoparametric. This will be fundamental in our subsequent complexity considerations.

Proposition 8 *Let $\mathcal{A}$ be an elementary routine of our branching–free computation model. Then $\mathcal{A}$ is output isoparametric. More explicitly, let β be a robust, parameterized arithmetic circuit with parameter domain $\mathcal{M}$. Suppose that β is an admissible input for $\mathcal{A}$. Let θ be a geometrically robust, constructible map defined on $\mathcal{M}$ such that θ represents the coefficient vector of the final results of β and let $\mathcal{T}$ be the image of θ. Then $\mathcal{T}$ is a constructible subset of a suitable affine space and there exists a geometrically robust, constructible map σ defined on $\mathcal{T}$ such that the composition map $\sigma \circ \theta$ represents the coefficient vector of the final results of $\mathcal{A}(\beta)$.*

A complete proof of this proposition is just tedious and will be omitted here. In case that $\mathcal{A}$ is a recursive routine, Proposition 8 expresses nothing but the requirement (B) applied to the output nodes of β.

Let assumptions and notations be as in Proposition 8 and suppose that there is given a (not necessarily convergent) sequence $(u_k)_{k\in\mathbb{N}}$ of parameter instances $u_k \in \mathcal{M}$ and that there exists a (possibly unknown) parameter instance $u \in \mathcal{M}$ such that the sequence $(\theta(u_k))_{k\in\mathbb{N}}$ converges to $\theta(u)$. In the spirit of [Ald84], [Lic90], §A and [BCS97] the sequence of (not necessarily consistent) ordinary arithmetic circuits $(\beta^{(u_k)})_{k\in\mathbb{N}}$ represents an *approximative algorithm* for the instantiation of the final results of β at u. From Theorem 5 we conclude that the constructible map σ

is strongly continuous and therefore the sequence $(\mathcal{A}(\beta)^{(u_k)})_{k\in\mathbb{N}}$ represents also an approximative algorithm for the instantiation of the final results of $\mathcal{A}(\beta)$ at u.

One sees easily that this property *characterizes* output parametricity of routines which are well behaved under restrictions.

Let us observe that Proposition 8 implies the following result.

Corollary 9 *Let assumptions and notations be as in Proposition 8. Then the routine $\mathcal{A}$ is output coalescent and satisfies the following condition:*

($*$) *Let u be an arbitrary parameter instance of $\mathcal{M}$ and let $\mathfrak{M}_u$ be the vanishing ideal of the $\mathbb{C}$–algebra $\mathbb{C}[\theta]$ at the point $\theta(u)$. Then the entries of the coefficient vector of the final results of $\mathcal{A}(\beta)$ are integral over the local $\mathbb{C}$–algebra $\mathbb{C}[\theta]_{\mathfrak{M}_u}$.*

The output coalescence of $\mathcal{A}$ and condition ($*$) are straight–forward consequences of the output isoparametricity of $\mathcal{A}$. We remark here that condition ($*$) follows already directly from the output coalescence of $\mathcal{A}$. This highlights again the close connection between isoparametricity and coalescence. The argument requires Zariski's Main Theorem. For details we refer to [CGH$^+$03], Sections 3.2 and 5.1.

3.3.3 The extended computation model

We are now going to extend our simplified branching–free computation model of elementary routines by a new model consisting of *algorithms* and *procedures* which may contain some limited branchings. Our description of this model will be rather informal. An algorithm will be a dynamic DAG of elementary routines which will be interpreted as pipes. At the end point of the pipes, decisions may be taken which depend on testing the validity of suitable universally quantified Boolean combinations of equalities between robust constructible functions defined on the parameter domain under consideration. The output of such an *equality test* is a bit vector which determines the next elementary routine (i.e., pipe) to be applied to the output circuit produced by the preceding elementary routine (pipe). This gives rise to a *extended computation model* which contains branchings. These branchings depend on a limited type of decisions at the level of the underlying abstract data type, namely the mentioned equality tests. We need to include this type of branchings in our extended computation model in order to capture the whole spectrum of known elimination procedures in effective algebraic geometry. Because of this limitation of branchings, we shall call the algorithms of our model *branching parsimonious* (compare [GH01] and [CGH$^+$03]). A branching parsimonious algorithm $\mathcal{A}$ which accepts a robust parameterized arithmetic circuit β with parameter domain $\mathcal{M}$ as input produces a new robust circuit $\mathcal{A}(\beta)$ with parameter domain $\mathcal{M}$. In particular $\mathcal{A}(\beta)$ *does not contain any branchings.*

Recall that our two main constructions of elementary routines depend on a previous selection of generic computations from a given shape list. This selection may be handled by calculations with the indexing of the shape list. We shall think that these calculations become realized by deterministic Turing machines. At the beginning, for a given robust parametric input circuit β with parameter domain $\mathcal{M}$, a tuple of fixed (i.e., of β independent) length of natural numbers is determined. This tuple constitutes an initial configuration of a Turing machine computation which determines the generic computations of our shape list that intervene in the elementary routine under construction. The entries of this tuple of natural numbers are called *invariants* of the circuit β. These invariants, whose values may also be Boolean (i.e., realized by the natural numbers 0 or 1), depend mainly on algebraic or geometric properties of the final results of β. However, they may also depend on structural properties of the labelled DAG β.

For example, the invariants of β may express that β has r parameters, n inputs and outputs, (over $\mathbb{C}$) non–scalar size and depth at most L and l, that β is totally division–free, that the final results of β have degree at most $d \leq 2^l$ and that for any parameter instance their specializations form a reduced regular sequence in $\mathbb{C}[X_1, \ldots, X_n]$, where $X_1, \ldots, X_n$ are the inputs of β.

Some of these invariants (e.g., the syntactical ones like number of parameters, inputs and outputs and non–scalar size and depth) may simply be read–off from the labelled DAG structure of β. Others, like the truth value of the statement that the specializations of final results of β in any parameter instance form a reduced regular sequence, have to be precomputed by an elimination algorithm from a previously given software library in effective commutative algebra or algebraic geometry or their value has to be fixed in advance as a precondition for the elementary routine which becomes applied to β.

In the same vein we may equip any elementary routine $\mathcal{A}$ with a Turing computable function which from the values of the invariants of a given input circuit β decides whether β is admissible for $\mathcal{A}$, and, if this is the case, determines the generic computations of our shape list which intervene in the application of $\mathcal{A}$ to β.

We shall now go a step further letting depend the internal structure of the computation on the circuit β. In the simplest case this means that we admit that the vector of invariants of β, denoted by $\mathrm{inv}(\beta)$, determines the architecture of a first elementary routine, say $\mathcal{A}_{\mathrm{inv}(\beta)}$, which admits β as input. Observe that the architectures of the elementary routines of our computation model may be characterized by tuples of fixed length of natural numbers. We consider this characterization as an *indexing* of the elementary routines of our computation model. We may now use this indexing in order to combine dynamically elementary routines by composition, join and union. Let us restrict our attention to the case of composition. In this case the output circuit of one elementary routine is the input for the next routine. The elementary routines which compose this display become implemented as pipes which start with a robust input circuit and end with a robust output circuit. Given such a pipe and an input circuit γ for the elementary routine $\mathcal{B}$ representing the pipe, we may apply suitable equality tests to the final results of $\mathcal{B}(\gamma)$ in order to determine a bit vector which we use to compute the index of the next elementary routine (seen as a new pipe) which will be applied to $\mathcal{B}(\gamma)$ as input.

A *low level program* of our extended computation model is now a text, namely the transition table of a deterministic Turing machine, which computes a function ψ realizing the following tasks.

Let as before β be a robust parameterized arithmetic circuit. Then ψ returns first on input $\mathrm{inv}(\beta)$ a Boolean value, zero or one, where one is interpreted as the informal statement "β is an admissible input". If this is the case, then ψ returns on $\mathrm{inv}(\beta)$ the index of an elementary routine, say $\mathcal{A}_{\mathrm{inv}(\beta)}$, which admits β as input. Then ψ determines the equality tests which have to be realized with the final results of $\mathcal{A}_{\mathrm{inv}(\beta)}(\beta)$. Depending on the outcome of these equality tests ψ determines an index value corresponding to a new elementary routine which admits $\mathcal{A}_{\mathrm{inv}(\beta)}(\beta)$ as input. Continuing in this way one obtains as end result an elementary routine $\mathcal{A}^{(\beta)}$, which applied to β, produces a final output circuit $\mathcal{A}^{(\beta)}(\beta)$. The function ψ represents all these index computations. We denote by $\psi(\beta)$ the *dynamic* vector of all data computed by ψ on input β.

The *algorithm* represented by ψ is the partial map between robust parametric arithmetic circuits that assigns to each admissible input β the circuit $\mathcal{A}^{(\beta)}(\beta)$ as output. Observe that elementary routines are particular algorithms. This kind of algorithms constitute our *extended computation model.* We remark that any algorithm of this model is *output isoparametric.* If the pipes of an algorithm are all represented by essentially division–free elementary routines, we call the algorithm itself *essentially division–free.*

One sees easily that the "Kronecker algorithm" [GLS01] (compare also [GHM$^+$98], [GHH$^+$97] and [GHMP97]) for solving non–degenerate polynomial equation systems over the complex numbers may be programmed in our extended computation model. Observe that the Kronecker algorithm requires more than a single elementary routine for its design. In order to understand this, recall that the Kronecker algorithm accepts as input an ordinary division–free arithmetic circuit which represents by its output nodes a reduced regular sequence of polynomials $G_1, \dots, G_n$ belonging to $\mathbb{C}[X_1, \dots, X_n]$. In their turn, the polynomials $G_1, \dots, G_n$ determine a *degree pattern*, say $\Delta := (\delta_1, \dots, \delta_n)$, with $\delta_i := \deg\{G_1 = 0, \dots, G_i = 0\}$ for $1 \leq i \leq n$.

After putting the variables $X_1, \dots, X_n$ in generic position with respect to $G_1, \dots, G_n$, the algorithm performs n recursive steps to eliminate them, one after the other. Finally the Kronecker algorithm produces an ordinary arithmetic circuit which computes the coefficients of $n + 1$ univariate polynomials $P, V_1, \dots, V_n$ over $\mathbb{C}$. These polynomials constitute a "geometric solution" (see [GLS01]) of the equation system $G_1 = 0, \dots, G_n = 0$ because they represent the zero dimensional algebraic variety $V := \{G_1 = 0, \dots, G_n = 0\}$ in the following "parameterized" form:

$$V := \{(V_1(t), \dots, V_n(t)); t \in \mathbb{C}, P(t) = 0\}.$$

Let β be any robust, parameterized arithmetic circuit with the same number of inputs and outputs, say $X_1, \dots, X_n$ and $G_1(U, X_1, \dots, X_n), \dots, G_n(U, X_1, \dots, X_n)$, respectively. Suppose that the parameter domain of β, say $\mathcal{M}$, is irreducible and that $\mathrm{inv}(\beta)$ expresses that for each parameter instance $u \in \mathcal{M}$ the polynomials $G_1(u, X_1, \ldots, X_n), \ldots, G_n(u, X_1, \ldots, X_n)$ form a reduced regular sequence in $\mathbb{C}[X_1, \ldots, X_n]$ with fixed (i.e., from $u \in \mathcal{M}$ independent) degree pattern. Suppose, furthermore, that the degrees of the individual polynomials $G_1(u, X_1, \ldots, X_n), \ldots,$ $G_n(u, X_1, \ldots, X_n)$ are also fixed. Then, on input β, the Kronecker algorithm runs a certain number (which depends on Δ) of elementary routines of our computation model which finally become combined by consistent iterative joins until the desired output is produced.

Another non–trivial example for an algorithm of our extended computation model, which involves only limited branchings, is the Gaussian elimination procedure of [Edm67] (or [Bar68]) applied to matrices whose entries are polynomials represented by ordinary arithmetic circuits in combination with a identity–to–zero test for such polynomials. The variables of these polynomials are considered as basic parameters and any admissible input circuit has to satisfy a certain precondition formulated as the non–vanishing of suitable minors of the given polynomial matrix. Details and applications of this type of Gaussian elimination for polynomial matrices can be found in [Hei83].

We say that a given algorithm $\mathcal{A}$ of our extended model *computes* (only) *parameters* if $\mathcal{A}$ satisfies the following condition:

for any admissible input β the final results of $\mathcal{A}(\beta)$ are all parameters.

Suppose that $\mathcal{A}$ is such an algorithm and β is the robust parametric arithmetic circuit with parameter domain $\mathcal{M}$ which we have considered before. Observe that $\mathcal{A}(\beta)$ contains the input variables $X_1, \ldots, X_n$ and that possibly new variables, which we call *auxiliary*, become introduced during the execution of the algorithm $\mathcal{A}$ on input β. Since the algorithm $\mathcal{A}$ computes only parameters, the input and auxiliary variables become finally eliminated by the application of recursive parameter routines and evaluations. We may therefore *collect garbage* in order to reduce $\mathcal{A}(\beta)$ to a *final output circuit* $\mathcal{A}_{\mathrm{final}}(\beta)$ whose intermediate results are only parameters.

If we consider the algorithm $\mathcal{A}$ as a partial map which assigns to each admissible input circuit β its final output circuit $\mathcal{A}_{\mathrm{final}}(\beta)$, we call $\mathcal{A}$ a *procedure.*

In this case, if ψ is a low level program defining $\mathcal{A}$, we call ψ a *low level procedure program.*

A particular feature of our extended computation model is the following:
there exists a non–negative integer f (depending on the recursion depth of $\mathcal{A}$) and non–decreasing real valued functions $C_f \geq 0$,$\ldots$, $C_0 \geq 0$ depending on one and the same dynamic integer vector, such that with the previous notations and L_β, $L_{\mathcal{A}(\beta)}$ denoting the non–scalar sizes of the circuits β and $\mathcal{A}(\beta)$ the condition

$$L_{\mathcal{A}(\beta)} \leq C_f(\psi(\beta))L_\beta^f + \cdots + C_0(\psi(\beta))$$

is satisfied.

In the case of the Kronecker algorithm (and most other elimination algorithms of effective Algebraic Geometry) we have $f := 1$, because the recursion depth of the basic routines which intervene is one.

In the sequel we shall need a particular variant of the notion of a procedure which enables us to capture the following situation.

Suppose we have to find a computational solution for a formally specified general algorithmic problem and that the formulation of the problem depends on certain parameter variables, say $U_1, \dots, U_r$, input variables, say $X_1, \dots, X_n$ and output variables, say $Y_1, \dots, Y_s$. Let such a problem formulation be given and suppose that its input is implemented by the robust parameterized arithmetic circuit β considered before, interpreting the parameter variables $U_1, \dots, U_r$ as the basic parameters $\pi_1, \dots, \pi_n$.

Then an algorithm $\mathcal{A}$ of our extended computation model which *solves* the given algorithmic problem should satisfy the architectural requirement we are going to describe now.

The algorithm $\mathcal{A}$ should be the composition of two subalgorithms $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ of our computation model which satisfy on input β the following conditions:

(i) *The subalgorithm $\mathcal{A}^{(1)}$ computes only parameters, β is admissible for $\mathcal{A}^{(1)}$ and none of the indeterminates $Y_1, \dots, Y_s$ is introduced in $\mathcal{A}^{(1)}(\beta)$ as auxiliary variable.*

(ii) *The circuit $\mathcal{A}^{(1)}_{\mathit{final}}(\beta)$ is an admissible input for the subalgorithm $\mathcal{A}^{(2)}$, the indeterminates $Y_1, \dots, Y_s$ occur as auxiliary variables in $\mathcal{A}^{(2)}(\mathcal{A}^{(1)}_{\mathit{final}}(\beta))$ and the final results of $\mathcal{A}^{(2)}(\mathcal{A}^{(1)}_{\mathit{final}}(\beta))$ depend only on $\pi_1, \dots, \pi_r$ and $Y_1, \dots, Y_s$ (all other auxiliary variables become eliminated during the execution of the subalgorithm $\mathcal{A}^{(2)}$ on the input circuit $\mathcal{A}^{(1)}_{\mathit{final}}(\beta)$).*

To the circuit $\mathcal{A}^{(2)}(\mathcal{A}^{(1)}_{\text{final}}(\beta))$ we may, as in the case when we compute only parameters, apply garbage collection. In this manner $\mathcal{A}^{(2)}(\mathcal{A}^{(1)}_{\text{final}}(\beta))$ becomes reduced to a final output circuit $\mathcal{A}_{\text{final}}(\beta)$ with parameter domain $\mathcal{M}$ which contains only the inputs $Y_1, \dots, Y_s$.

Observe that the subalgorithm $\mathcal{A}^{(1)}$ is by Proposition 8 an output isoparametric procedure of our extended computation model (the same is also true for the subalgorithm $\mathcal{A}^{(2)}$, but this will not be relevant in the sequel).

We consider the algorithm $\mathcal{A}$, as well as the subalgorithms $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$, as *procedures* of our extended computation model. In case that the *subprocedures* $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ are essentially division–free, we call also the procedure $\mathcal{A}$ *essentially division–free*. This will be of importance in Section 4.

The architectural requirement given by conditions (i) and (ii) may be interpreted as follows:
the subprocedure $\mathcal{A}^{(1)}$ is a pipeline which transmits only parameters to the subprocedure $\mathcal{A}^{(2)}$. In particular, no (true) rational function is transmitted from $\mathcal{A}^{(1)}$ to $\mathcal{A}^{(2)}$.

Nevertheless, let us observe that on input β the procedure $\mathcal{A}$ establishes by means of the underlying low level program ψ an additional link between β and the subprocedure $\mathcal{A}^{(2)}$ applied to the input $\mathcal{A}^{(1)}(\beta)$. The elementary routines which constitute $\mathcal{A}^{(2)}$ on input $\mathcal{A}^{(1)}(\beta)$ become determined by index computations which realizes ψ on $\mathrm{inv}(\beta)$ and certain equality tests between the intermediate results of $\mathcal{A}^{(1)}(\beta)$. In this sense the subprocedure $\mathcal{A}^{(1)}$ transmits not only parameters to the subprocedure but also a limited amount of digital information which stems from the input circuit β.

The decomposition of the procedure $\mathcal{A}$ into two subprocedures $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ satisfying conditions (i) and (ii) represents an architectural restriction which is justified when it makes sense to require that on input β the number of essential additions and multiplications contained in $\mathcal{A}_{\text{final}}(\beta)$ is bounded by a function which depends only on $\mathrm{inv}(\beta)$. In Section 4.1, we shall make a substantial use of this restriction and give such a justification in the particular case of elimination algorithms.

Here, we shall only point out the following consequence of this restriction. Let assumptions and notations be as before, let G, ν and F be vectors composed by the final results of β, $\mathcal{A}^{(1)}(\beta)$ and $\mathcal{A}_{\text{final}}(\beta)$, respectively, and let θ and φ be the coefficient vectors of G and F. Then the images of θ and ν are constructible subsets $\mathcal{T}$ and $\mathcal{T}'$ of suitable affine spaces and there exist geometrically robust constructible maps σ and σ' defined on $\mathcal{T}$ and $\mathcal{T}'$ with $\nu = \sigma \circ \theta$ and $\varphi = \sigma' \circ \nu = \sigma' \circ \sigma \circ \theta$.

Based on [HK04] and [GHKa], we shall develop in future work a high level specification language for algorithms and procedures of our computation model. The idea is to use a generalized variant of the *extended constraint data base model* introduced in [HK04] in order to specify algorithmic problems in symbolic Scientific Computing, especially in effective algebraic geometry (e.g., effective elimination problems; see Section 4). In this sense the procedure $\mathcal{A}$, which solves the algorithmic problem considered before, will turn out to be a *query computation* composed by two subprocedures namely $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ which compute each a subquery of the query which specifies the given algorithmic problem. All these queries are called *geometric* because the procedures $\mathcal{A}^{(1)}$, $\mathcal{A}^{(2)}$ and $\mathcal{A}$ are output isoparametric (see [GHKa]).

4 Applications of the extended computation model to complexity issues of effective elimination theory

In this section we shall always work with procedures of our extended, branching parsimonious computation model. We shall study representative examples of elimination problems in effective algebraic geometry which certify, to a different extent, that *branching parsimonious* elimination procedures based on our computation paradigm *cannot run in polynomial time.*

4.1 Flat families of zero–dimensional elimination problems

We first introduce, in terms of abstract data types, the notion of a flat family of zero–dimensional elimination problems (see also [GH01] and [CGH$^+$03]). Then we fix the classes of (concrete) objects, namely robust parameterized arithmetic circuits with suitable parameter domains, which represent ("implement") these problems by means of a suitable abstraction function.

Throughout this section, we suppose that there are given indeterminates $U_1, \dots, U_r$, $X_1, \dots, X_n$ and Y over $\mathbb{C}$.

As concrete objects we shall consider robust parameterized arithmetic input and output circuits with parameter domain $\mathbb{A}^r$. The indeterminates $U_1, \dots, U_r$ will play the role of the basic parameters. The input nodes of the input circuits will be labelled by $X_1, \dots, X_n$, whereas the output circuits will have a single input node, labelled by Y.

Let us now define the meaning of the term "flat family of zero–dimensional elimination problems" (in the basic parameters $U_1, \dots, U_r$ and the inputs $X_1, \dots, X_n$). Let $U := (U_1, \dots, U_r)$ and $X := (X_1, \dots, X_n)$ and let $G_1, \dots, G_n$ and H be polynomials belonging to the $\mathbb{C}$–algebra $\mathbb{C}[U, X] := \mathbb{C}[U_1, \dots, U_r, X_1, \dots, X_n]$. Suppose that the polynomials $G_1, \dots, G_n$ form a regular sequence in $\mathbb{C}[U, X]$, thus defining an equidimensional subvariety $V := \{G_1 = 0, \dots, G_n = 0\}$ of the $(n + r)$–dimensional affine space $\mathbb{A}^r \times \mathbb{A}^n$. The algebraic variety V has dimension r. Let δ be the (geometric) degree of V (observe that this degree does not take into account multiplicities or components at infinity). Suppose, furthermore, that the morphism of affine varieties $\pi : V \to \mathbb{A}^r$, induced by the canonical projection of $\mathbb{A}^r \times \mathbb{A}^n$ onto $\mathbb{A}^r$, is finite and generically unramified (this implies that π is flat and that the ideal generated by $G_1, \dots, G_n$ in $\mathbb{C}[U, X]$ is radical). Let $\tilde{\pi} : V \to \mathbb{A}^{r+1}$ be the morphism defined by $\tilde{\pi}(v) := (\pi(v), H(v))$ for any point v of the variety V. The image of $\tilde{\pi}$ is a hypersurface of $\mathbb{A}^{r+1}$ whose minimal equation is a polynomial of $\mathbb{C}[U, Y] := \mathbb{C}[U_1, \dots, U_r, Y]$ which we denote by F. Let us write $\deg F$ for the total degree of the polynomial F and $\deg_Y F$ for its partial degree in the variable Y. Observe that F is monic in Y and that $\deg F \leq \delta \deg H$ holds. Furthermore, for a Zariski dense set of points u of $\mathbb{A}^r$, we have that $\deg_Y F$ is the cardinality of the image of the restriction of H to the finite set $\pi^{-1}(u)$. The polynomial $F(U, H)$ vanishes on the variety V.

Let us consider an arbitrary point $u := (u_1, \dots, u_r)$ of $\mathbb{A}^r$. For given polynomials $A \in \mathbb{C}[U, X]$ and $B \in \mathbb{C}[U, Y]$, we denote by $A^{(u)}$ and $B^{(u)}$ the polynomials $A(u_1, \dots, u_r, X_1, \dots, X_n)$ and $B(u_1, \dots, u_r, Y)$ which belong to $\mathbb{C}[X] := \mathbb{C}[X_1, \dots, X_n]$ and $\mathbb{C}[Y]$ respectively. Similarly we denote for an arbitrary polynomial $C \in \mathbb{C}[U]$ by $C^{(u)}$ the value $C(u_1, \dots, u_r)$ which belongs to the field $\mathbb{C}$. The polynomials $G_1^{(u)}, \dots, G_n^{(u)}$ define the zero–dimensional subvariety

$$V^{(u)} := \left\{ G_1^{(u)} = 0, \dots, G_n^{(u)} = 0 \right\} \cong \pi^{-1}(u)$$

of the affine space $\mathbb{A}^n$. The degree (i.e., the cardinality) of $V^{(u)}$ is bounded by δ. Denote by $\tilde{\pi}^{(u)} : V^{(u)} \to \mathbb{A}^1$ the morphism induced by the polynomial $H^{(u)}$ on the variety $V^{(u)}$. Observe that the polynomial $F^{(u)}$ vanishes on the (finite) image of the morphism $\tilde{\pi}^{(u)}$. Observe also that the polynomial $F^{(u)}$ is not necessarily the minimal equation of the image of $\tilde{\pi}^{(u)}$.

We call the equation system $G_1 = 0, \ldots, G_n = 0$ and the polynomial H a *flat family of zero–dimensional elimination problems depending on the basic parameters $U_1, \ldots, U_r$ and the inputs $X_1, \ldots, X_n$* and we call F the associated *elimination polynomial.* A point $u \in \mathbb{A}^r$ is considered as a *parameter instance* which determines a *particular problem instance*, consisting of the equations $G_1^{(u)} = 0, \ldots, G_n^{(u)} = 0$ and the polynomial $H^{(u)}$. A power of the polynomial $F^{(u)}$ is called a *solution* of this particular problem instance.

The equation system $G_1 = 0, \ldots, G_n = 0$ together with the polynomial H is also called the *general instance* of the given flat family of elimination problems and any power of the elimination polynomial F is also called a *general solution* of this flat family.

We suppose now that the general instance of the given flat family of elimination problems is implemented by an essentially division–free, robust parameterized arithmetic circuit β with parameter domain $\mathbb{A}^r$ and inputs $X_1, \ldots, X_n$, whose final results are the polynomials $G_1, \ldots, G_n$ and H. The task is to find another essentially division–free, robust parameterized arithmetic circuit γ with parameter domain $\mathbb{A}^r$ having a single output node, labelled by Y, which computes for a suitable integer $q \in \mathbb{N}$ the power F^q of the polynomial F. We suppose, furthermore, that this goal is achieved by the application of an essentially division–free procedure $\mathcal{A}$ of our extended computation model to the input circuit β. Thus we may put $\gamma := \mathcal{A}_{\text{final}}(\beta)$ and γ may be interpreted as an essentially division–free circuit over $\mathbb{C}[U]$ with a single input Y (observe that the parameters computed by the robust circuits β, $\mathcal{A}(\beta)$ and $\mathcal{A}_{\text{final}}(\beta)$ are geometrically robust constructible functions with domain of definition $\mathbb{A}^r$ which belong by [GHMS11], Corollary 12 to the $\mathbb{C}$–algebra $\mathbb{C}[U]$). Using the geometric properties of flat families of zero–dimensional problems, we deduce from [GHM$^+$98], [GHH$^+$97],[GHMP97], [GLS01] or alternatively from [CGH89], [DFGS91] that such essentially division–free procedures always exist and that they compute even the elimination polynomial F (the reader may notice that one needs for this argument the full power of our computation model which includes divisions by parameters).

We say that the essentially division–free procedure $\mathcal{A}$ *solves algorithmically* the general instance of the given flat family of zero–dimensional elimination problems.

From now on we suppose that there is given a procedure $\mathcal{A}$ of our extended computation model, decomposed in two essentially division–free subprocedures $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ as in Section 3.3.3, such that $\mathcal{A}$ solves algorithmically the general instance of any given flat family of zero–dimensional elimination problems. Our circuit β is therefore an admissible input for $\mathcal{A}$ and hence for $\mathcal{A}^{(1)}$. The final results of $\mathcal{A}^{(1)}(\beta)$ constitute a geometrically robust constructible map ν defined on $\mathbb{A}^r$ which represents by means of $\mathcal{A}^{(1)}_{\text{final}}(\beta)$ an admissible input for the procedure $\mathcal{A}^{(2)}$. Moreover, $\gamma := \mathcal{A}_{\text{final}}(\beta)$ is an essentially division–free parameterized arithmetic circuit with parameter domain $\mathbb{A}^r$ and input Y.

Let $\mathcal{S}$ be the image of the geometrically robust constructible map ν. Then $\mathcal{S}$ is an irreducible constructible subset of a suitable affine space. Analyzing now the internal structure of the essentially division–free, robust parameterized arithmetic circuit $\mathcal{A}^{(2)}(\mathcal{A}^{(1)}(\beta))$, one sees easily that there exists a geometrically robust constructible map ψ defined on $\mathcal{S}$ such that the entries of the geometrically robust composition map $\nu^* := \psi \circ \nu$ constitute the essential parameters of the circuit γ. Let m be the number of components of the map ν^*. Since ν and ν^* are composed by geometrically robust constructible functions defined on $\mathbb{A}^r$, we deduce from [GHMS11], Corollary 12 that ν and ν^* may be interpreted as vectors of polynomials of $\mathbb{C}[U]$.

The circuit γ is essentially division–free. Hence there exists a vector ω of m–variate polynomials over $\mathbb{C}$ such that the polynomials of $\mathbb{C}[U]$, which constitute the entries of $\omega(\nu^*)$, become the coefficients of the elimination polynomial F with respect to the main indeterminate Y (see [KP96], Section 2.1). Observe that we may write $\omega(\nu^*) = \omega \circ \nu^*$ interpreting the entries of ν^* as polynomials of $\mathbb{C}[U]$.

We are now going to see what happens at a particular parameter instance $u \in \mathbb{A}^r$. Since β, $\mathcal{A}^{(1)}(\beta)$, $\mathcal{A}(\beta)$ and $\gamma = \mathcal{A}_{\text{final}}(\beta)$ are essentially division–free, robust parameterized arithmetic circuits with parameter domain $\mathbb{A}^r$, we may specialize the vector U of basic parameters to the parameter instance $u \in \mathbb{A}^r$, obtaining thus ordinary division–free arithmetic circuits over $\mathbb{C}$ with the same inputs. We denote them by the superscript u, namely by $\beta^{(u)}$, $(\mathcal{A}^{(1)}(\beta))^{(u)}$, $(\mathcal{A}(\beta))^{(u)}$ and $\gamma^{(u)}$. One sees immediately that $G_1^{(u)}, \dots, G_n^{(u)}$ and $H^{(u)}$ are the final results of $\beta^{(u)}$, that the entries of $\nu(u)$ are the final results of $(\mathcal{A}^{(1)}(\beta))^{(u)}$ and that $(F^{(u)})^q$ is the final result of $\mathcal{A}(\beta)^{(u)}$ and $\gamma^{(u)}$. Observe that the division–free circuit $\gamma^{(u)}$ uses only the entries of $\nu^*(u)$ and fixed rational numbers as scalars.

In the same spirit as before, we say that the procedure $\mathcal{A}$ solves algorithmically the particular instance, which is determined by u, of the given flat family of zero–dimensional elimination problems.

Let us here clarify how all this is linked to the rest of the terminology used in [CGH$^+$03]. In this terminology the polynomial map given by ω defines a "holomorphic encoding" of the set of solutions of all particular problem instances and $\nu^*(u)$ is a "code" of the particular solution $(F^{(u)})^q$. In the same context the robust constructible map ν^* is called an "elimination procedure" which is "robust" since the procedure $\mathcal{A}^{(1)}$ is output isoparametric and since ν^* is geometrically robust (compare [CGH$^+$03], Definition 5, taking into account Lemma 7, Proposition 8 and Corollary 9 above).

In this sense, we speak about *families* of zero–dimensional elimination problems and their instances and not simply about a single (particular or general) zero–dimensional elimination problem.

Let us now turn back to the discussion of the given essentially division–free procedure $\mathcal{A}$ which solves algorithmically the general instance of any flat family of zero–dimensional elimination problems.

We are now going to show the main result of this section, namely that the given procedure $\mathcal{A}$ *cannot run in polynomial time.*

Theorem 10 *Let notations and assumptions be as before. For any natural number n there exists an essentially division–free, robust parameterized arithmetic circuit β_n with basic parameters T, $U_1, \ldots, U_n$ and inputs $X_1, \ldots, X_n$ which for $U := (U_1, \ldots, U_n)$ and $X := (X_1, \ldots, X_n)$ computes polynomials $G_1^{(n)}, \ldots, G_n^{(n)} \in \mathbb{C}[X]$ and $H^{(n)} \in \mathbb{C}[T, U, X]$ such that the following conditions are satisfied:*

(i) The equation system $G_1^{(n)} = 0, \ldots, G_n^{(n)} = 0$ and the polynomial $H^{(n)}$ constitute a flat family of zero–dimensional elimination problems, depending on the parameters T, $U_1, \ldots, U_n$ and the inputs $X_1, \ldots, X_n$, with associated elimination polynomial $F^{(n)} \in \mathbb{C}[T, U, Y]$.

(ii) β_n is an ordinary division–free arithmetic circuit of size $O(n)$ over $\mathbb{C}$ with inputs T, $U_1, \ldots, U_n$, $X_1, \ldots, X_n$.

(iii) $\gamma_n := \mathcal{A}_{final}(\beta_n)$ is an essentially division–free robust parameterized arithmetic circuit with basic parameters $T, U_1, \ldots, U_n$ and input Y such that γ_n computes for a suitable integer $q_n \in \mathbb{N}$ the polynomial $(F^{(n)})^{q_n}$. The circuit γ_n performs at least $\Omega(2^{\frac{n}{2}})$ essential multiplications and at least $\Omega(2^n)$ multiplications with parameters. Therefore γ_n has, as ordinary arithmetic circuit over $\mathbb{C}$ with inputs $T, U_1, \ldots, U_n, X_1, \ldots, X_n$, non–scalar size at least $\Omega(2^n)$.

Proof. During our argumentation we shall tacitly adapt to the new context the notations introduced before. We shall follow the main technical ideas behind the papers [GH01], [CGH⁺03] and [GHMS11]. We fix now the natural number n and consider the polynomials

$$G_1 := G_1^{(n)} := X_1^2 - X_1, \ldots, G_n := G_n^{(n)} := X_1^2 - X_n$$

and

$$H := H^{(n)} := \sum_{1 \leq i \leq n} 2^{i-1} X_i + T \prod_{1 \leq i \leq n} (1 + (U_i - 1) X_i)$$

which belong to $\mathbb{C}[X]$ and to $\mathbb{C}[T, U, X]$, respectively.

Observe that $G_1, \ldots, G_n$ and H may be evaluated by a division–free ordinary arithmetic circuit $\beta := \beta_n$ over $\mathbb{C}$ which has non–scalar size $O(n)$ and inputs T, $U_1, \ldots, U_n$, $X_1, \ldots, X_n$. As parameterized arithmetic circuit β is therefore robust. Hence β satisfies condition (ii) of the theorem.

One sees easily that $G_1 = 0, \ldots, G_n = 0$ and H constitute a flat family of zero–dimensional elimination problems depending on the parameters T, $U_1, \ldots, U_n$ and the inputs $X_1, \ldots, X_n$.

Let us write H as a polynomial in the main indeterminates $X_1, \ldots, X_n$ with coefficients $\theta_{\kappa_1, \ldots, \kappa_n} \in \mathbb{C}[T, U]$, $\kappa_1, \ldots, \kappa_n \in \{0, 1\}$, namely

$$H = \sum_{\kappa_1, \ldots, \kappa_n \in \{0,1\}} \theta_{\kappa_1, \ldots, \kappa_n} X_1^{\kappa_1}, \ldots, X_n^{\kappa_n}.$$

Observe that for $\kappa_1, \ldots, \kappa_n \in \{0, 1\}$ the polynomial $\theta_{\kappa_1,\ldots,\kappa_n}(0, U) \in \mathbb{C}[U]$ is of degree at most zero, i.e., a constant complex number, independent of $U_1, \ldots, U_n$.

Let $\theta := (\theta_{\kappa_1,\ldots,\kappa_n})_{\kappa_1,\ldots,\kappa_n \in \{0,1\}}$ and observe that the vector $\theta(0, U)$ is a fixed point of the affine space $\mathbb{A}^{2^n}$. We denote by $\mathfrak{M}$ the vanishing ideal of the $\mathbb{C}$–algebra $\mathbb{C}[\theta]$ at this point.

Consider now the polynomial

$$F := F^{(n)} := \prod_{0 \leq j \leq 2^n - 1} (Y - (j + T \prod_{1 \leq i \leq n} U_i^{[j]_i}))$$

of $\mathbb{C}[T, U, Y]$, where $[j]_i$ denotes the i–th digit of the binary representation of the integer j, $0 \leq j \leq 2^{n-1} - 1$, $1 \leq i \leq n$. Let $q := q_n$.

One sees easily that F is the elimination polynomial associated with the given flat family of zero–dimensional elimination problems $G_1 = 0, \ldots, G_n = 0$ and H.

Let us write F^q as a polynomial in the main indeterminate Y with coefficients $\varphi_\kappa \in \mathbb{C}[T, U]$, $1 \leq \kappa \leq 2^n q$, namely

$$F^q = Y^{2^n q} + \varphi_1 Y^{2^n q - 1} + \cdots + \varphi_{2^n q}.$$

Observe that for $1 \leq \kappa \leq 2^n q$ the polynomial $\varphi_\kappa(0, U) \in \mathbb{C}[U]$ is of degree at most zero. Let $\lambda_\kappa := \varphi_\kappa(0, U)$, $\lambda := (\lambda_\kappa)_{1 \leq \kappa \leq 2^n q}$ and $\varphi := (\varphi_\kappa)_{1 \leq \kappa \leq 2^n q}$. Observe that λ is a fixed point of the affine space $\mathbb{A}^{2^n q}$.

Recall that β is an admissible input for the procedure $\mathcal{A}$ and hence for $\mathcal{A}^{(1)}$, that the final results of $\mathcal{A}^{(1)}(\beta)$ constitute the entries of the robust constructible map ν defined on $\mathbb{A}^{n+1}$, that ν represents an admissible input for the procedure $\mathcal{A}^{(2)}$ and that $\gamma = \mathcal{A}_{\text{final}}(\beta)$ is an essentially division–free, parameterized arithmetic circuit with parameter domain $\mathbb{A}^{n+1}$ and input Y.

Furthermore, recall that there exists a geometrically robust constructible map ψ defined on the image $\mathcal{S}$ of ν such that the entries of $\nu^* = \psi \circ \nu$ constitute the essential parameters of the circuit γ, that the entries of ν and ν^* may be interpreted as polynomials of $\mathbb{C}[T, U]$ and that for m being the number of components of the map ν^*, there exists a vector ω of m–variate polynomials over $\mathbb{C}$ such that the polynomials of $\mathbb{C}[T, U]$ which constitute the entries of $\omega(\nu^*) = \omega \circ \nu^*$ become the coefficients of the polynomial F^q with respect to the main indeterminate Y. Let $\mathcal{T}$ be the image of the coefficient vector θ of H, and interpret θ as a geometrically robust constructible map defined on $\mathbb{A}^{n+1}$. Observe that $\mathcal{T}$ is a constructible subset of $\mathbb{A}^{2^n}$. Since H is the unique final result of the circuit β, we deduce from Proposition 8 that there exists a geometrically robust constructible map σ defined on $\mathcal{T}$ satisfying the condition $\nu = \sigma \circ \theta$. This implies $\nu^* = \psi \circ \sigma \circ \theta$ and, following [GHMS11], Corollary 12 and Definition 6 (i), that the entries of ν^* are polynomials of $\mathbb{C}[T, U]$ which are integral over the local $\mathbb{C}$–subalgebra $\mathbb{C}[\theta]_{\mathfrak{M}}$ of $\mathbb{C}(T, U)$.

Let $\mu \in \mathbb{C}[T, U]$ be such an entry. Then there exists an integer s and polynomials $a_0, a_1, \ldots, a_s \in \mathbb{C}[\theta]$ with $a_0 \notin \mathfrak{M}$ such that the algebraic dependence relation

$$a_0 \mu^s + a_1 \mu^{s-1} + \cdots + a_s = 0 \tag{3}$$

is satisfied in $\mathbb{C}[T,U]$. From (3) we deduce the algebraic dependence relation

$$a_0(0,U)\mu(0,U)^s + a_1(0,U)\mu(0,U)^{s-1} + \cdots + a_s(0,U) = 0 \tag{4}$$

in $\mathbb{C}[U]$.

Since the polynomials $a_0, a_1, \ldots, a_s$ belong to $\mathbb{C}[\theta]$ and $\theta(0,U)$ is a fixed point of $\mathbb{A}^{2^n}$, we conclude that $\alpha_0 := a_0(0,U), \alpha_1 := a_1(0,U), \ldots, \alpha_s := a_s(0,U)$ are complex numbers. Moreover, $a_0 \notin \mathfrak{M}$ implies $\alpha_0 \neq 0$.

Thus (4) may be rewritten into the algebraic dependence relation

$$\alpha_0\mu(0,U)^s + \alpha_1\mu(0,U)^{s-1} + \cdots + \alpha_s = 0 \tag{5}$$

in $\mathbb{C}[U]$ with $\alpha_0 \neq 0$.

This implies that the polynomial $\mu(0,U)$ of $\mathbb{C}[U]$ is of degree at most zero.

Therefore $w := \nu^*(0,U)$ is a fixed point of the affine space $\mathbb{A}^m$. Since γ computes the polynomial F^q and F^q has the form $F^q = Y^{2^n q} + \varphi_1 Y^{2^n q-1} + \cdots + \varphi_{2^n q}$ with $\varphi_\kappa \in \mathbb{C}[T,U]$, $1 \leq \kappa \leq 2^n q$, we see that $\varphi = (\varphi_\kappa)_{1\leq\kappa\leq 2^n q}$ may be decomposed as follows:

$$\varphi = \omega(\nu^*) = \omega \circ \nu^*.$$

Recall that $\lambda = (\lambda_\kappa)_{1\leq\kappa\leq 2^n q}$ with $\lambda_\kappa := \varphi_\kappa(0,U)$, $1 \leq \kappa \leq 2^n q$, is a fixed point of the affine space $\mathbb{A}^{2^n}$.

For $1 \leq \kappa \leq 2^n q$ we may write the polynomial $\varphi_\kappa \in \mathbb{C}[T,U]$ as follows:

$$\varphi_\kappa = \lambda_\kappa + \Delta_\kappa T + \text{ terms of higher degree in } T \tag{6}$$

with $\Delta_\kappa \in \mathbb{C}[U]$. From [CGH$^+$03], Lemma 6 we deduce that the elimination polynomial F has the form $F = Y^{2^n} + B_1 Y^{2^n-1} + \cdots + B_{2^n}$, where for $1 \leq l \leq 2^n$ the coefficient B_l is an element of $\mathbb{C}[T,U]$ of the form

$$B_l = (-1)^l \sum_{l\leq j_1<\cdots<j_l<2^n} j_1 \ldots j_l + TL_l + \text{ terms of higher degree in } T,$$

where $L_1, \ldots, L_{2^n} \in \mathbb{C}[U]$ are $\mathbb{C}$–linearly independent.

Choose now different complex numbers $\eta_1, \ldots, \eta_{2^n}$ from $\mathbb{C} - \{j \in \mathbb{Z}; 0 \leq j < 2^n\}$ and observe that for $1 \leq \kappa' \leq 2^n$ the identities

$$\frac{\partial F^q}{\partial T}(0,U,\eta_{\kappa'}) = qF^{q-1}(0,U,\eta_{\kappa'})\frac{\partial F}{\partial T}(0,U,\eta_{\kappa'}) = q \prod_{0\leq j<2^n} (\eta_{\kappa'} - j)^{q-1} \sum_{1\leq l\leq 2^n} L_l \eta_{\kappa'}^{2^n-l}$$

and

$$\frac{\partial F^q}{\partial T}(0,U,\eta_{\kappa'}) = \sum_{1\leq\kappa\leq 2^n q} \Delta_\kappa \eta_{\kappa'}^{2^n q-\kappa}$$

hold.

Since $L_1, \ldots, L_{2^n}$ are $\mathbb{C}$–linearly independent, we deduce from the non–singularity of the Vandermonde matrix $(\eta_{\kappa'}^{2^n-l})_{0 \leq l, \kappa' \leq 2^n}$ that 2^n many of the polynomials $\Delta_1, \ldots, \Delta_{2^n q}$ of $\mathbb{C}[U]$ are $\mathbb{C}$–linearly independent.

Consider now an arbitrary point $u \in \mathbb{A}^n$ and let $\epsilon_u : \mathbb{A}^1 \to \mathbb{A}^m$ and $\delta_u : \mathbb{A}^1 \to \mathbb{A}^{2^n q}$ be the polynomial maps defined for $t \in \mathbb{A}^1$ by $\epsilon_u(t) := \nu^*(t, u)$ and $\delta_u(t) := \varphi(t, u)$. Then we have $\epsilon_u(0) = \nu^*(0, u) = w$ and $\delta_u(0) = \varphi(0, u) = \lambda$, independently of u. Moreover, from $\varphi = \omega \circ \nu^*$ we deduce $\delta_u = \omega \circ \epsilon_u$.

Thus (6) implies

$$(\Delta_1(u), \ldots, \Delta_{2^n q}(u)) = \frac{\partial \varphi}{\partial t}(0, u) = \delta'_u(0) = (D\omega)_w(\epsilon'_u(0)), \tag{7}$$

where $(D\omega)_w$ denotes the (first) derivative of the m–variate polynomial map ω at the point $w \in \mathbb{A}^m$ and $\delta'_u(0)$ and $\epsilon'_u(0)$ are the derivatives of the parameterized curves δ_u and ϵ_u at the point $0 \in \mathbb{A}^1$. We rewrite now (7) in matrix form, replacing $(D\omega)_w$ by the corresponding transposed Jacobi matrix $M \in \mathbb{A}^{m \times 2^n q}$ and $\delta'_u(0)$ and $\epsilon'_u(0)$ by the corresponding points of $\mathbb{A}^{2^n q}$ and $\mathbb{A}^m$, respectively.

Then (7) takes the form

$$(\Delta_1(u), \ldots, \Delta_{2^n q}(u)) = \epsilon'_u(0) M, \tag{8}$$

where the complex $(m \times 2^n q)$–matrix M is independent of u.

Since 2^n many of the polynomials $\Delta_1, \ldots, \Delta_{2^n} \in \mathbb{C}[U]$ are $\mathbb{C}$–linearly independent, we may choose points $u_1, \ldots, u_{2^n} \in \mathbb{A}^n$ such that the complex $(2^n \times 2^n q)$–matrix

$$N := (\Delta_\kappa(u_l))_{\substack{1 \leq l \leq 2^n \\ 1 \leq \kappa \leq 2^n q}}$$

has rank 2^n.

Let K be the complex $(2^n \times m)$–matrix whose rows are $\epsilon'_{u_1}(0), \ldots, \epsilon'_{u_{2^n}}(0)$.

Then (8) implies the matrix identity

$$N = K \cdot M.$$

Since N has rank 2^n, the rank of the complex $(m \times 2^n)$–matrix M is at least 2^n. This implies

$$m \geq 2^n. \tag{9}$$

Therefore the circuit γ contains $m \geq 2^n$ essential parameters.

Let L be the number of essential multiplications executed by the parameterized arithmetic circuit γ and let L' be the total number of multiplications of γ, excepting those by scalars from $\mathbb{C}$. Then, after a well–known standard rearrangement [PS73] of γ, we may suppose without loss of generality, that there exists a constant $c > 0$ (independent of the input circuit γ and the procedure $\mathcal{A}$) such that $L \geq cm^{\frac{1}{2}}$ and $L' \geq cm$ holds.

From the estimation (9) we deduce now that the circuit γ performs at least $\Omega(2^{\frac{n}{2}})$ essential multiplications and at least $\Omega(2^n)$ multiplications, including also multiplications with parameters. This finishes the proof of the theorem. ■

Observation Let assumptions and notations be as before. In the proof of Theorem 10 we made a substantial use of the output isoparametricity of the procedure $\mathcal{A}^{(1)}$ when we applied Proposition 8 in order to guarantee the existence of a geometrically robust constructible map σ defined on $\mathcal{T}$ which satisfies the condition $\nu = \sigma \circ \theta$. The conclusion was that the entries of $\nu^* = \psi \circ \nu$ are polynomials of $\mathbb{C}[T, U]$ which are integral over $\mathbb{C}[\theta]_{\mathfrak{M}}$. This implied finally that $\nu^*(0, U)$ is a fixed point of the affine space $\mathbb{A}^m$. Taking into account the results of [CGH$^+$03], Sections 3.2 and 5.1 it suffices to require that the procedure $\mathcal{A}^{(1)}$ is *output coalescent* in order to arrive to the same conclusion. This means that Theorem 10 remains valid if we require only that the procedure $\mathcal{A}^{(1)}$ is output coalescent.

In the proof of Theorem 10 we have exhibited an infinite sequence of flat families of zero–dimensional elimination problems represented by robust parameterized arithmetic circuits of small size, such that any implementation of their associated elimination polynomials, obtained by a procedure of our extended computation model which solves the given elimination task for any instance, requires circuits of exponential size.

The statement of Theorem 10 may also be interpreted in terms of a mathematically certified trade–off of quality attributes. Suppose for the moment that we had built our model for branching parsimonious computation in the same way as in Section 3.3, omitting the requirement (B) for recursive routines, however. Recall that this requirement implies the output isoparametricity of any algorithm of our extended computation model and recall from Section 3.3.2 that well behavedness under reduction is a quality attribute which implies output isoparametricity and therefore also the conclusion of Theorem 10.

A complexity class like “exponential time in worst case” represents also a quality attribute. Thus we see that the quality attribute “well behavedness under reduction” implies the quality attribute “exponential time in worst case” for any essentially division–free procedure of our extended computation model which solves algorithmically the general instance of any given flat family of zero–dimensional problems.

The proof of Theorem 10 depends substantially on the decomposition of the elimination procedure $\mathcal{A}$ into two subprocedures $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ satisfying conditions (i) and (ii) of Section 3.3.3. We are now going to justify this architectural restriction on the procedure $\mathcal{A}$ for the particular case of elimination algorithms.

As at the beginning of this section, let $U := (U_1, \dots, U_r)$, $X := (X_1, \dots, X_n)$, $G_1, \dots, G_n$, $H \in \mathbb{C}[U, X]$ and $F \in \mathbb{C}[U, Y]$ such that $G_1 = 0, \dots, G_n = 0$ and H constitute a flat family of zero–dimensional elimination problems and F its associated elimination polynomial. Suppose that $G_1, \dots, G_n$ and H are implemented by an essentially division–free, robust parameterized arithmetic circuit β with parameter domain $\mathbb{A}^r$ and inputs $X_1, \dots, X_n$.

All *known* algorithms which solve the general instance of any flat family of zero–dimensional elimination problems may be interpreted as belonging to our restricted set of procedures. They compute directly the elimination polynomial F (and not an arbitrary power of it). Thus let $\mathcal{A}$ be such a known algorithm and let $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ be the subalgorithms which compose $\mathcal{A}$ in the same way as before. Then $\mathcal{A}^{(1)}$ computes the coefficients of F, where F is considered as a polynomial over $\mathbb{C}[U]$ in the indeterminate Y. The subalgorithm $\mathcal{A}^{(2)}$ may be interpreted as the Horner scheme which evaluates F from its precomputed coefficients and Y.

Observe that F, and hence $\deg_Y F$, depends only on the polynomials $G_1, \ldots, G_n$ and H, but not on the particular circuit β. Therefore $\deg_Y F$ is determined by $\psi(\beta)$, where ψ is the low level program of the algorithm $\mathcal{A}$.
For any parameter instance $u \in \mathbb{A}^r$ we may think $(\mathcal{A}^{(1)}(\beta))^{(u)}$ as a constraint database (in the sense of [HK04] and [GHKa]) which allows to evaluate the univariate polynomial $F^{(u)} \in \mathbb{C}[Y]$ as often as desired for arbitrary inputs $y \in \mathbb{A}^1$, using each time a number of arithmetic operations in $\mathbb{C}$, namely $\deg_Y F$, which does not depend on the non–scalar size of β.

Moreover $\mathcal{A}$ satisfies the following condition:

(D) *There exist non–decreasing real valued functions* $C_1 \geq 0$ *and* $C_2 \geq 0$ *depending on dynamic integer vectors, such that for* L_β *and* $L_{\mathcal{A}(\beta)}$*, being the non–scalar sizes of the circuits* β *and* $\mathcal{A}(\beta)$*, the inequality*

$$L_{\mathcal{A}(\beta)} \leq C_1(\psi(\beta))L_\beta + C_2(\psi(\beta))$$

holds.

Let now $\mathcal{A}$ be an *arbitrary*, essentially division–free algorithm of our extended computation model which solves the general instance of any flat family of zero–dimensional elimination problems and let β be an input circuit for $\mathcal{A}$ which represents a particular family of such problems. Let F be the associated elimination polynomial.

Then the complexity of the algorithm $\mathcal{A}$ is only competitive with known elimination algorithms if we require that the number of *essential* additions and multiplications of $\mathcal{A}_{\text{final}}(\beta)$ is bounded by $2 \cdot \deg_Y F$. This leads us to the requirement that $\mathcal{A}$ must be decomposable in two subalgorithms $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$ as above.

Therefore any elimination algorithm of our extended computation model which is claimed to improve upon known algorithms for *all* admissible input circuits β, must have this architectural structure. In particular, such an algorithm cannot call the input circuit β when the output variable Y became already involved. This justifies the architectural restriction we made in the statement and proof of Theorem 10.

Moreover, the competitivity of $\mathcal{A}$ with known elimination algorithms requires that $\mathcal{A}$ must satisfy condition (D).

From Theorem 10 and its proof we deduce now the lower bound

$$\max\{C_1(\psi(\beta_n)), C_2(\psi(\beta_n))\} = \Omega(\frac{\delta_n}{L_{\beta_n}}),$$

where δ_n is the geometric degree of the subvariety of $\mathbb{A}^r \times \mathbb{A}^{n+1}$ defined by the polynomials $G_1^{(n)}, \ldots, G_n^{(n)}, Y - H^{(n)}$ (observe $\delta_n = 2^n$). Adding to β_n a suitable addition node we obtain a totally division–free new circuit β_n^* which represents $G_1^{(n)}, \ldots, G_n^{(n)}$ and $Y - H^{(n)}$. Observe that for each $(s,u) \in \mathbb{A}^1 \times \mathbb{A}^n$ the degree pattern of the polynomials $G_1^{(n)}, \ldots, G_n^{(n)}, Y - H(s,u,X)$ is constant and the system degree is δ_n. The polynomial $F^{(n)}$ is the output of the Kronecker algorithm applied to β_n^* and the variable Y. Therefore the algorithm $\mathcal{A}$ produces on β_n the same output as the Kronecker algorithm applied to β_n^* and the variable Y. We conclude now from $L_{\beta_n^*} = O(n)$ that the Kronecker algorithm is nearly optimal in our extended computation model.

In our computation model, algorithms are transformations of parameterized arithmetic circuits over one and the same parameter domain. This represents a substantial ingredient for the proof of Theorem 10. If we allow branchings which lead to subdivisions of the parameter domain of the input circuit, the conclusion of Theorem 10 may become uncertain (see [GHKb]).

4.2 The elimination of a block of existential quantifiers

Let notations be the same as in the proof of Theorem 10 in Section 4.1. Let $n \in \mathbb{N}$, $S_1, \ldots, S_n$ new indeterminates, $S := (S_1, \ldots, S_n)$, $\hat{G}_1^{(n)} := X_1^2 - X_1 - S_1, \ldots, \hat{G}_n^{(n)} := X_n^2 - X_n - S_n$ and again $H^{(n)} := \sum_{1 \le i \le n} 2^{i-1} X_i + T \prod_{1 \le i \le n} (1 + (U_i - 1)X_i)$.

Observe that the polynomials $\hat{G}_1^{(n)}, \ldots, \hat{G}_n^{(n)}$ form a reduced regular sequence in $\mathbb{C}[S,T,U,X]$ and that they define a subvariety $\hat{V}_n$ of the affine space $\mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n \times \mathbb{A}^n$ which is isomorphic to $\mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n$ and hence irreducible and of dimension $2n+1$. Moreover, the morphism $\hat{V}_n \to \mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n$ which associates to any point $(s,t,u,x) \in \hat{V}_n$ the point (s,t,u), is finite and generically unramified. Therefore the morphism $\hat{\pi}_n : \hat{V}_n \to \mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n \times \mathbb{A}^1$ which associates to any $(s,t,u,x) \in \hat{V}_n$ the point $(s,t,u,H^{(n)}(t,u,x)) \in \mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n \times \mathbb{A}^1$ is finite and its image $\hat{\pi}_n(\hat{V}_n)$ is a hypersurface of $\mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n \times \mathbb{A}^1$ with irreducible minimal equation $\hat{F}^{(n)} \in \mathbb{C}[S,T,U,Y]$.

Hence $\hat{G}_1^{(n)} = 0, \ldots, \hat{G}_n^{(n)} = 0$ and $H^{(n)}$ represent a flat family of zero–dimensional elimination problems whose associated elimination polynomial is just $\hat{F}^{(n)}$.

Observe that $\deg \hat{F}^{(n)} = \deg_Y \hat{F}^{(n)} = 2^n$ and that for $0 \in \mathbb{A}^n$ the identity

$$\hat{F}^{(n)}(0,T,U,Y) = F^{(n)}(T,U,Y) \text{ holds,}$$

where $F^{(n)}$ is the elimination polynomial associated with the flat family of zero dimensional elimination problems given by $X_1^2 - X_1 = 0, \ldots, X_n^2 - X_n = 0$ and $H^{(n)}$. Since $\hat{F}^{(n)}$ is irreducible, any equation of $\mathbb{C}[S,T,U,Y]$ which defines $\hat{\pi}_n(\hat{V}_n)$ in $\mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n \times \mathbb{A}^1$ is without loss of generality a power of $\hat{F}^{(n)}$.

We consider now $S_1, \ldots, S_n, T, U_1, \ldots, U_n$ as basic parameters, $X_1, \ldots, X_n$ as input and Y as output variables.

Let $\mathcal{A}'$ be an essentially division–free procedure of our extended computation model satisfying the following condition:
$\mathcal{A}'$ accepts as input any robust parameterized arithmetic circuit β which represents

the general instance of a flat family of zero–dimensional elimination problems with associated elimination polynomial F and $\mathcal{A}'_{\rm final}(\beta)$ has a single input Y and a single final result which defines the same hypersurface as F.

With this notions and notations we have the following result.

Proposition 11 *There exist an ordinary division–free arithmetic circuit $\hat{\beta}_n$ of size $O(n)$ over $\mathbb{C}$ with inputs $S_1, \ldots, S_n$, T, $U_1, \ldots, U_n$, $X_1, \ldots, X_n$ and final results $\hat{G}_1^{(n)}, \ldots, \hat{G}_n^{(n)}, H^{(n)}$. The essentially division–free, robust parameterized arithmetic circuit $\hat{\gamma}_n := \mathcal{A}'_{\mathit{final}}(\hat{\beta}_n)$ depends on the basic parameters $S_1, \ldots, S_n$, T, $U_1, \ldots, U_n$ and the input Y and its single final result is a power of $\hat{F}^{(n)}$. The circuit $\hat{\gamma}_n$ performs at least $\Omega(2^{\frac{n}{2}})$ essential multiplications and at least $\Omega(2^n)$ multiplications with parameters. As ordinary arithmetic circuit over $\mathbb{C}$ with inputs $S_1, \ldots, S_n$, T, $U_1, \ldots, U_n$ and Y, the circuit $\hat{\gamma}_n$ has non–scalar size at least $\Omega(2^n)$.*

Proof. The existence of an ordinary division–free arithmetic circuit as in the statement of Proposition 11 is evident. The rest follows immediately from the proof of Theorem 10 in Section 4.1 by restricting the parameter domain $\mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n$ of $\hat{\beta}_n$ to $\mathbb{A}^1 \times \mathbb{A}^n$, i.e., by specializing S to $0 \in \mathbb{A}^n$. ■

Suppose now that there is given another essentially division–free procedure $\mathcal{A}''$ of our extended computation model satisfying the following condition:
$\mathcal{A}''$ accepts as input any robust arithmetic circuit β which represents the general instance of a flat family of zero–dimensional elimination problems with associated elimination polynomial F and there exists a Boolean circuit b in as many variables as the number of final results of $\mathcal{A}''_{\rm final}(\beta)$ such that the algebraic variety defined by F coincides with the constructible set which can be described by plugging into b the final results of $\mathcal{A}''_{\rm final}(\beta)$ as polynomial equations.

Observe that this represents the most general architecture we can employ to adapt in the spirit of Section 3.3.3 our extended computation model for *functions* to *parametric decision problems.*

Let $s \in \mathbb{N}$ and $A_1, \ldots, A_s$ new indeterminates with $A := (A_1, \ldots, A_s)$. We suppose that there is given an essentially division–free procedure $\mathcal{B}$ of our extended computation model which accepts as input any essentially division–free, robust parameterized arithmetic circuit γ with the basic parameters $A_1, \ldots, A_s$ and the input variable Y, such that $\mathcal{B}_{\rm final}(\gamma)$ represents, by its output nodes, in $\mathbb{C}[A, Y]$ the multiplicative decomposition of the final results of γ by their greatest common divisor and complementary factors.

In this sense, we call the procedure $\mathcal{B}$ a *GCD algorithm.*

Let $\psi_{\mathcal{A}''}$ and $\psi_{\mathcal{B}}$ be the given low level programs of the procedures $\mathcal{A}''$ and $\mathcal{B}$. We require that $\mathcal{A}''$ and $\mathcal{B}$ are competitive with known algorithms which solve the same tasks. Following our argumentation in Section 4.1 we may therefore suppose that there exist four non–decreasing real valued functions $C_1 \geq 0$, $C_2 \geq 0$ and $D_1 \geq 0$,

$D_2 \geq 0$ which depend on dynamic integer vectors and which satisfy the estimates

$$L_{\mathcal{A}''(\beta)} \leq C_1(\psi_{\mathcal{A}''}(\beta))L_\beta + C_2(\psi_{\mathcal{A}''}(\beta))$$

and

$$L_{\mathcal{B}(\gamma)} \leq D_1(\psi_{\mathcal{B}}(\gamma))L_\gamma + D_2(\psi_{\mathcal{B}}(\gamma)).$$

We consider again the ordinary division–free arithmetic circuit $\hat{\beta}_n$ of Proposition 8 which represents the polynomials $\hat{G}_1^{(n)}, \ldots, \hat{G}_n^{(n)}$ and $H^{(n)}$.

With these notions and notations we may now formulate the following statement.

Theorem 12 *Let assumptions and notations be as before. Then we have*

$$max\{C_i(\psi_{\mathcal{A}''}(\hat{\beta}_n)), D_i(\psi_{\mathcal{B}}(\mathcal{A}''_{final}(\hat{\beta}_n))); i = 1,2\} = \Omega(\frac{2^{\frac{n}{2}}}{n})$$

Proof. If we plug into the Boolean circuit b the final results of $\mathcal{A}''_{\text{final}}(\hat{\beta}_n)$ as polynomial equations, we obtain by assumption a description of the hypersurface $\hat{\pi}(\hat{V}_n)$ of the affine space $\mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n \times \mathbb{A}^1$. This implies that between the final results of $\mathcal{A}''_{\text{final}}(\hat{\beta}_n)$ there exists a selection, say the polynomials $P_1, \ldots, P_m$ and $R_1, \ldots, R_t$ of $\mathbb{C}[S, T, U, Y]$ such that the formula

$$P_1 = 0 \wedge \cdots \wedge P_m = 0 \wedge R_1 \neq 0 \wedge \cdots \wedge R_t \neq 0$$

defines a nonempty Zariski open (and dense) subset of the irreducible surface $\hat{\pi}(\hat{V}_n)$ of $\mathbb{A}^n \times \mathbb{A}^1 \times \mathbb{A}^n \times \mathbb{A}^1$.

Let $R := R_1 \ldots R_t$ and observe that the greatest common divisor of $P_1, \ldots, P_m$ has the form $(\hat{F}^{(n)})^q \cdot Q$, where q belongs to $\mathbb{N}$ and Q is the greatest common divisor of $P_1, \ldots, P_m, R$. Therefore we may compute $(F^{(n)})^q$ in the following way: erasing suitable nodes from the circuit $\mathcal{A}''_{\text{final}}(\hat{\beta}_n)$ and adding $t-1$ multiplication nodes we obtain two robust parameterized arithmetic circuits $\gamma_1^{(n)}$ and $\gamma_2^{(n)}$ with basic parameters $S_1, \ldots, S_n$, T, $U_1, \ldots, U_n$ and input Y whose final results are $P_1, \ldots, P_m$ and $P_1, \ldots, P_m, R$ respectively.

Between the final results of $\mathcal{B}_{\text{final}}(\gamma_1^{(n)})$ and $\mathcal{B}_{\text{final}}(\gamma_2^{(n)})$ are the polynomials $(\hat{F}^{(n)})^q \cdot Q$ and Q. Applying the procedure $\mathcal{B}$ to the union of $\mathcal{B}_{\text{final}}(\gamma_1^{(n)})$ and $\mathcal{B}_{\text{final}}(\gamma_2^{(n)})$ we obtain finally an essentially division–free, robust parameterized arithmetic circuit with basic parameters $S_1, \ldots, S_n, T, U_1, \ldots, U_n$ and input Y whose single final result is $(\hat{F}^n)^q$.

Joining the circuits $\mathcal{A}''(\hat{\beta}_n)$, $\mathcal{B}_{\text{final}}(\gamma_1^{(n)})$, $\mathcal{B}_{\text{final}}(\gamma_2^{(n)})$ and the final division node we obtain an ordinary arithmetic circuit of non–scalar size at most

$$1 + 3L_{\mathcal{B}(\mathcal{A}''_{\text{final}}(\hat{\beta}_n))} \leq$$

$$1 + 3(D_1(\psi_{\mathcal{B}}(\mathcal{A}''_{\text{final}}(\hat{\beta}_n)))L_{\mathcal{A}''_{\text{final}}(\hat{\beta}_n)} + D_2(\psi_{\mathcal{B}}(\mathcal{A}''_{\text{final}}(\hat{\beta}_n)))) \leq$$

$$1+3C_1(\psi_{\mathcal{A}}(\hat{\beta}_n))D_1(\psi_{\mathcal{B}}(\mathcal{A}''_{\text{final}}(\hat{\beta}_n)))L_{\hat{\beta}_n}+$$
$$3C_2(\psi_{\mathcal{A}}(\hat{\beta}_n))D_1(\psi_{\mathcal{B}}(\mathcal{A}''_{\text{final}}(\hat{\beta}_n)))+D_2(\psi_{\mathcal{B}}(\mathcal{A}''_{\text{final}}(\hat{\beta}_n))).$$

On the other hand we deduce from Theorem 10

$$L_{\hat{\beta}_n}=O(n) \text{ and } 1+3L_{\mathcal{B}(\mathcal{A}''_{\text{final}}(\hat{\beta}_n))}=\Omega(2^n).$$

This implies the estimate of Theorem 12. ■

In a simple minded understanding, Theorem 12 says that in our extended computation model either the elimination of a single existential quantifier block in a prenex first–order formula of the elementary language of $\mathbb{C}$ or the computation of a greatest common divisor of a finite set of circuit represented polynomials requires *exponential time*. Complexity results in this spirit were already obtained in [GH01] and [CGH$^+$03] (compare also Proposition 11 and Observation in Section 4.1).

4.3 Arithmetization techniques for Boolean circuits

Let $m \in \mathbb{N}$ and let $0, 1$ and $Z_1, \dots, Z_m$ be given constants and variables. Let $Z := (Z_1, \dots, Z_m)$. Following the context we shall interpret $0, 1$ as Boolean values or the corresponding complex numbers and $Z_1, \dots, Z_m$ as Boolean variables or indeterminates over $\mathbb{C}$. With $\wedge, \vee, ^-$ we denote the Boolean operations "and", "or" and "not". A Boolean circuit b with inputs $Z_1, \dots, Z_m$ is a DAG whose indegree zero nodes are labelled by $0, 1$ and $Z_1, \dots, Z_m$ and whose inner nodes have indegree two or one. In the first case an inner node is labelled by $\wedge$ or $\vee$ and in the second by $^-$. Some inner nodes of b become labelled as outputs. We associate with b a semantics as follows:

- indegree zero nodes which are labelled by $0, 1$ become interpreted by the corresponding constant functions $\{0,1\}^m \to \{0,1\}$,
- indegree zero nodes which are labelled by $Z_1, \dots, Z_m$ become interpreted by the corresponding projection function $\{0,1\}^m \to \{0,1\}$,
- let ρ be an inner node of b of indegree two whose parent nodes ρ_1 and ρ_2 are already interpreted by Boolean functions $g_{\rho_1}, g_{\rho_2} : \{0,1\}^m \to \{0,1\}$. If ρ is labelled by $\wedge$, we interpret ρ by the Boolean function $g_\rho := g_{\rho_1} \wedge g_{\rho_2}$ and if ρ is labelled by $\vee$, we interpret ρ by the Boolean function $g_\rho := g_{\rho_1} \vee g_{\rho_2}$,
- let ρ be an inner node of b of indegree one whose parent node ρ' became already interpreted by a Boolean function $g_{\rho'} : \{0,1\}^m \to \{0,1\}$. Then we interpret ρ by the Boolean function $g_\rho := \overline{g_{\rho'}}$.

For a node ρ of b we call g_ρ the *intermediate result* of b at ρ. If ρ is an output node, we call g_ρ a *final result* of b.

An arithmetization β of the Boolean circuit b consists of the same DAG as b with a different labelling as follows.

Let U, V be new indeterminates over $\mathbb{C}$. The constants $0, 1$ become interpreted by the correspondent complex numbers and $Z_1, \ldots, Z_m$ as indeterminates over $\mathbb{C}$. Let ρ be an inner node of β. If ρ has indegree two, then ρ becomes labelled by a polynomial $R_\rho \in \mathbb{Z}[U, V]$ and if ρ has indegree one by a polynomial $R_\rho \in \mathbb{Z}[U]$. The output nodes of β and b are the same.

Representing for each inner node ρ of β the polynomial G_ρ by a division–free ordinary arithmetic circuit over $\mathbb{Z}$ in the inputs U, V or U, we obtain an ordinary division–free arithmetic circuit over $\mathbb{Z}$ in the inputs $Z_1, \ldots, Z_m$.

Just as we did in Section 3.3.2 we may associate with β a semantics which determines for each node ρ of β a polynomial $G_\rho \in \mathbb{Z}[Z]$. We say that β is an *arithmetization* of the Boolean circuit b if the following condition is satisfied:
for any node ρ of b and any argument $z \in \{0, 1\}^m$ the Boolean value $g_\rho(z)$ coincides with the arithmetic value $G_\rho(z)$ (in a somewhat imprecise notation: $g_\rho(z) = G_\rho(z)$).

An example of an arithmetization procedure is given by the map which associates to each node ρ of b a polynomial $[g_\rho]$ of $\mathbb{Z}[Z]$ satisfying the following conditions:

- $[0] = 0$, $[1] = 1$, $[Z_1] = Z_1, \ldots, [Z_m] = Z_m$

- for ρ an inner node of indegree two of b with parents ρ_1 and ρ_2:

$$[g_\rho] = [g_{\rho_1}] \cdot [g_{\rho_2}] \text{ if the label of } \rho \text{ is } \wedge$$

  and

$$[g_\rho] = [g_{\rho_1}] + [g_{\rho_2}] - [g_{\rho_1}] \cdot [g_{\rho_2}] \text{ if the label of } \rho \text{ is } \vee$$

- for ρ an inner node of indegree one of b with parent ρ':

$$[g_\rho] = 1 - [g_{\rho'}].$$

The resulting arithmetic circuit is called the *standard arithmetization* of b (see, e.g., [Sha92] and [BF91]).

Let $n, r \in \mathbb{N}$ and $U_1, \ldots, U_r, X_1, \ldots, X_n$ be new variables. For $m := n + r$ we replace now Z by U and X, where $U := (U_1, \ldots, U_r)$ and $X := (X_1, \ldots, X_n)$. We shall interpret $U_1, \ldots, U_r$ as parameters and $X_1, \ldots, X_n$ as input variables.

Let b be a Boolean circuit with the inputs $U_1, \ldots, U_r, X_1, \ldots, X_n$ and just one final result $h : \{0, 1\}^r \times \{0, 1\}^n \to \{0, 1\}$.

We wish to describe the set of instances $u \in \{0, 1\}^r$ where $h(u, X_1, \ldots, X_n)$ is a satisfiable Boolean function.

For this purpose let us choose an arithmetization β of b. We interpret β as an ordinary arithmetic circuit over $\mathbb{Z}$ with the parameters $U_1, \ldots, U_r$ and the inputs $X_1, \ldots, X_n$. The single final result of β is a polynomial $H \in \mathbb{Z}[U, X]$ which satisfies for any $u \in \{0, 1\}^r$, $x \in \{0, 1\}^n$ the following condition:

$$h(u, x) = H(u, x).$$

Without loss of generality we may suppose that the polynomials $X_1^2 - X_1$, …, $X_n^2 - X_n$ are intermediate results of β. We relabel now β such that these polynomials and H become the final results of β. Observe that $X_1^2 - X_1 = 0, \dots, X_n^2 - X_n = 0$ and H represent a flat family of zero–dimensional elimination problems.

Let Y be a new indeterminate and let $F \in \mathbb{Z}[U, Y]$ the associated elimination polynomial. One verifies easily the identity

$$F(U, Y) = \prod_{x \in \{0,1\}^n} (Y - H(U, x)).$$

Let $\mathcal{A}$ be an essentially division–free procedure of our extended computation model which solves algorithmically the general instance of any flat family of zero–dimensional elimination problems. Then β is an admissible input for $\mathcal{A}$ and there exists an integer $q \in \mathbb{N}$ such that F^q is the final result of $\mathcal{A}_{\text{final}}(\beta)$.

We consider now the task to count for any $u \in \{0,1\}^r$ the number k of instances $x \in \{0,1\}^n$ with $h(u, x) = 1$.

The polynomial F^q encodes two possible solutions of this task.

The first solution is the following: let l be the order of the univariate polynomial $F^q(u, Y)$ at zero. Then q divides l and we have $k = 2^n - \frac{l}{q}$.

The second and more interesting solution is the following: write $F^q = Y^{2^n q} + \varphi_1 Y^{2^n q - 1} + \cdots + \varphi_{2^n q}$ with $\varphi_1, \dots, \varphi_{2^n q} \in \mathbb{Z}[U]$. Then $\varphi_1(u)$ is an integer which is divisible by q and we have $k = -\frac{\varphi_1(u)}{q}$.

Observe also $\deg \varphi_1 \leq \deg_U H$.

These considerations show the relevance of an *efficient* evaluation of the polynomial F^q (e.g., by the circuit $\mathcal{A}_{\text{final}}(\beta)$).

We ask therefore whether $\mathcal{A}_{\text{final}}(\beta)$ can be polynomial in the size of the Boolean circuit b. The following example illustrates that the answer may become negative.

In the sequel we are going to exhibit for $r := 2n + 1$ a Boolean circuit b of size $O(n)$ which evaluates a function $h : \{0,1\}^r \times \{0,1\}^n \longrightarrow \{0,1\}$ such that the standard arithmetization β of b represents a flat family of zero–dimensional elimination problems with associated elimination polynomial F and such that any essentially division–free procedure $\mathcal{A}$ of our extended computation model that accepts the input β and computes by means of $\mathcal{A}_{\text{final}}(\beta)$ a power of F, requires time $\Omega(2^n)$ for this task. This means that it is unlikely that algorithms designed following the paradigm of object–oriented programming are able to evaluate the polynomial φ_1 efficiently.

On the other hand, since the degree of φ_1 is bounded by $\deg_U H$ and therefore "small", there exists a polynomial time interactive protocol which checks for any $u \in \{0,1\}^r$ and any $c \in \mathbb{Z}$ the equation $\varphi_1(u) = c$. Thus this problem belongs to the complexity class IP (see [LFKN92] for details).

We are now going to exhibit an example of a Boolean circuit which highlights the unfeasibility of our computation task.

For this purpose let $r := 2n + 1$ and $S_1, \dots, S_n, T, U_1, \dots, U_n$ parameters and $X_1, \dots, X_n$ input variables and let $S := (S_1, \dots, S_n)$ and $U := (U_1, \dots, U_n)$.

We consider the Boolean function $h : \{0,1\}^{2n+1} \times \{0,1\}^n \to \{0,1\}$ defined by the Boolean formula

$$\phi := \bigwedge_{1\le i\le n} (\overline{X_i} \vee (S_i \wedge X_i)) \vee (T \wedge \bigwedge_{1\le i\le n} (\overline{X_i} \vee (U_i \wedge X_i))).$$

From the structure of the formula ϕ we infer that h can be evaluated by a Boolean circuit b of size $O(n)$ in the inputs $S_1, \dots, S_n, T, U_1, \dots, U_n$.

Let β be the standard arithmetization of the Boolean circuit b and let H be the final result of β. Observe that the total, and hence the non–scalar size of β is $O(n)$. Then we have

$$H = \prod_{1\le i\le n} (1 + (S_i - 1)X_i) + (1 - \prod_{1\le i\le n} (1 + (S_i - 1)X_i))T \prod_{1\le i\le n} (1 + (U_i - 1)X_i).$$

Observe that the equations $X_1^2 - X_1 = 0, \dots, X_n^2 - X_n = 0$ and the polynomial H represent a flat family of zero–dimensional elimination problems. Let F be the associated elimination polynomial. Then F can be written as

$$F = Y^{2^n} + B_1 Y^{2^n - 1} + \cdots + B_{2^n} = \prod_{0\le j<2^n} (Y - (\prod_{1\le i\le n} S^{[j]_i} + (1 - \prod_{1\le j\le n} S^{[j]_i})T \prod_{1\le i\le n} U_i^{[j]_i}))$$

with

$$B_k = (-1)^k \sum_{0\le j_1<\cdots<j_k<2^n} \prod_{1\le h\le k} (\prod_{1\le i\le n} S_i^{[j_h]_i} + (1 - \prod_{1\le i\le n} S_i^{[j_h]_i})T \prod_{1\le i\le n} U_i^{[j_h]_i})$$

for $1 \le k \le 2^n$.

Let

$$L_k := (-1)^k \sum_{0\le j_1<\cdots<j_k<2^n} \sum_{1\le h\le k} \prod_{1\le i\le n} S_i^{[j_1]_i} \dots (1 - \prod_{1\le i\le n} S_i^{[j_h]_i}) \cdots \prod_{1\le i\le n} S_i^{[j_k]_i} \prod_{1\le i\le n} U_i^{[j_h]_i},$$

where $1 \le k \le 2^n$.

Then we have

$$B_k = (-1)^k \sum_{0\le j_1<\cdots<j_k<2^n} \prod_{1\le i\le n} S_i^{[j_1]_i} \cdots \prod_{1\le i\le n} S_i^{[j_k]_i} + L_k.T + \text{ terms of higher degree in } T$$

Let $\epsilon : \mathbb{A}^{2^n} \to \mathbb{A}^{2^n}$ be the morphism of affine spaces which assigns to each point $z \in \mathbb{A}^{2^n}$ the values of the elementary symmetric functions in 2^n variables at z. Observe that the Jacobian of ϵ at $(\prod_{1\le i\le n} S^{[j]_i})_{0\le j<2^n}$ is a non–singular $(2^n \times 2^n)$–matrix $N(S)$. The polynomials $L_k, 1 \le k \le 2^n$ are obtained by applying $N(S)$ to $((1 - \prod_{1\le i\le n} S^{[j]_i}) \prod_{1\le i\le n} U_i^{[j]_i})_{0\le j<2^n}$. Since the monomials $\prod_{1\le i\le n} U_i^{[j]_i}, 0 \le j < 2^n$, are linearly independent over $\mathbb{C}(S)$ we conclude that the polynomials $L_k, 1 \le k \le 2^n$ have the same property.

With this preparation we are now able to repeat textually the arguments in the proof of Theorem 10 of Section 4.1 in order to show the following statement.

Theorem 13 *Let assumptions and notations be as before and let $\mathcal{A}$ be an essentially division free procedure of our extended computation model which accepts the arithmetic circuit β as input. Suppose that $\mathcal{A}_{\mathit{final}}(\beta)$ has a unique final result and that it is a power of the elimination polynomial F. Then the non–scalar size of $\mathcal{A}_{\mathit{final}}(\beta)$ is at least $\Omega(2^n)$.*

4.4 Divisions and blow ups

We are now going to analyze the main argument of the proof of Theorem 10 from a geometric point of view.

We recall first some notations and assumptions we made during this proof.

With respect to the indeterminates $X_1, \ldots, X_n$, we considered the vector θ of coefficients of the expression

$$H = \sum_{1 \le i \le n} 2^{i-1} X_i + T \prod_{1 \le i \le n} (1 + (U_i - 1) X_i)$$

as a polynomial map $\mathbb{A}^{n+1} \to \mathbb{A}^{2^n}$ with image $\mathcal{T}$. Recall that $\mathcal{T}$ is an irreducible constructible subset of $\mathbb{A}^{2^n}$.

Further, with respect to the indeterminate Y, we considered the vector φ of nontrivial coefficients of the monic polynomial

$$F = \prod_{1 \le j \le 2^n - 1} (Y - (j + T \prod_{1 \le i \le n} U_i^{[j]_i}))$$

also as a polynomial map $\mathbb{A}^{n+1} \to \mathbb{A}^{2^n}$.

One sees immediately that there exists a unique polynomial map $\eta : \mathcal{T} \to \mathbb{A}^{2^n}$ such that $\varphi = \eta \circ \theta$ holds. Using particular properties of θ and φ we showed implicitly in the proof of Theorem 10 that η satisfies the following condition:

> *Let m be a natural number, $\zeta : \mathcal{T} \to \mathbb{A}^m$ a geometrically robust constructible and $\pi : \mathbb{A}^m \to \mathbb{A}^{2^n}$ a polynomial map such that $\eta = \pi \circ \zeta$ holds. Then the condition*
>
> $$m \ge 2^n$$
>
> *is satisfied.*

This means that the following computational task cannot be solved efficiently:

Allowing certain restricted divisions, reduce the datum θ consisting of 2^n entries to a datum ζ consisting of only $m \le 2^n$ entries such that the vector η still may be recovered from ζ without using any division, i.e., an ordinary division–free arithmetic circuit over $\mathbb{C}$.

Here the allowed divisions involve only quotients which are geometrically robust functions defined on $\mathcal{T}$.

In order to simplify the following discussion we shall assume without loss of generality that all our constructible maps have geometrically robust extensions to $\overline{\mathcal{T}}$.

Let f and g be two elements of the coordinate ring $\mathbb{C}[\overline{\mathcal{T}}]$ of the affine variety $\overline{\mathcal{T}}$ and suppose that $g \neq 0$ holds and that the element $\frac{f}{g}$ of the rational function field $\mathbb{C}(\overline{\mathcal{T}})$ may be extended to a robust constructible function defined on $\overline{\mathcal{T}}$, which we denote also by $\frac{f}{g}$, since this extension is unique.

Then we have two cases: the coordinate function g divides f in $\mathbb{C}[\overline{\mathcal{T}}]$ or not. In the first case we may compute $\frac{f}{g}$, by means of an ordinary division–free arithmetic circuit over $\mathbb{C}$, from the restrictions to $\overline{\mathcal{T}}$ of the canonical projections $\mathbb{A}^{2^n} \to \mathbb{A}^1$. Thus $\frac{f}{g}$ belongs to the coordinate ring $\mathbb{C}[\overline{\mathcal{T}}]$. In the second case this is not anymore true and $\mathbb{C}[\overline{\mathcal{T}}][\frac{f}{g}]$ is a proper extension of $\mathbb{C}[\overline{\mathcal{T}}]$ in $\mathbb{C}(\overline{\mathcal{T}})$. In both cases $\mathbb{C}[\overline{\mathcal{T}}][\frac{f}{g}]$ is the coordinate ring of an affine chart of the blow up of $\mathbb{C}[\overline{\mathcal{T}}]$ at the ideal generated by f and g. We refer to this situation as a *division blow up* which we call *essential* if $\frac{f}{g}$ does not belong to $\mathbb{C}[\overline{\mathcal{T}}]$.

Therefore we have shown in the proof of Theorem 10 that essential division blow ups do not help to solve efficiently the reduction task formulated before.

A similar situation arises in multivariate polynomial interpolation (see [GHMS11], Theorem 23).

Following [Har92], Theorem 7.2.1 any rational map may be decomposed into a finite sequence of successive blow ups followed by a regular morphism of algebraic varieties. Our method indicates the interest to find lower bounds for the number of blow ups (and their embedding dimensions) necessary for an effective variant of this result.

Problem adapted methods for proving lower bounds for the number of blow ups necessary to resolve singularities would also give indications which order of complexity can be expected for efficient desingularization algorithms (see [EV00]). At this moment only upper bound estimations are known [Bla09].

4.5 Comments on complexity models for geometric elimination

The question, whether $P \neq NP$ or $P_{\mathbb{C}} \neq NP_{\mathbb{C}}$ holds in the classical or the BSS Turing Machine setting, concerns only computational *decision* problems. These, on their turn, are closely related to the task to construct *efficiently*, for a given prenex existential formula, an equivalent, quantifier free one (compare [BSS89], [HM93], [SS95] and [BCSS98]). In the sequel we shall refer to this and to similar, geometrically motivated computational tasks as "*effective elimination*".

Theorem 10 in Section 4.1 does not establish a fact concerning decision problems like the $P_{\mathbb{C}} \neq NP_{\mathbb{C}}$ question. It deals with the *evaluation of a function* which assigns to suitable prenex existential formulas over $\mathbb{C}$ *canonical*, equivalent and quantifier–free formulas of the same elementary language.

Theorem 10 says that in our computation model this function cannot be evaluated efficiently. If we admit also *non–canonical* quantifier–free formulas as function values (i.e., as outputs of our algorithms), then this conclusion remains true, provided that the calculation of parameterized greatest common divisors is feasible and efficient in our model (see [CGH$^+$03], Section 5).

It is not clear what this implies for the $P_{\mathbb{C}} \neq NP_{\mathbb{C}}$ question.

Intuitively speaking, our exponential lower complexity bound for effective geometric elimination is only meaningful and true for computer programs designed in a professional way by software engineers. Hacker programs are excluded from our considerations.

This constitutes an enormous difference between our approach and that of Turing machine based complexity models, which always include the hacker aspect. Therefore the proof of a striking lower bound for effective elimination becomes difficult in these models.

Our argumentation is based on the requirement of output parametricity which on its turn is the consequence of two other requirements, a functional and a non–functional one, that we may employ alternatively. More explicitly, we require that algorithms (and their specifications) are described by branching parsimonious asserted programs or, alternatively, that they behave well under reductions (see Sections 3.3.2 and 3.3.3).

Let us observe that the complexity statement of Theorem 10 refers to the relationship between input and output and not to a particular property of the output alone. In particular, Theorem 10 does not imply that certain polynomials, discussed below, like the permanent or the Pochhammer polynomials, are hard to evaluate.

Let notations and assumptions be as in Section 4.1. There we considered for arbitrary $n \in \mathbb{N}$ the flat family of zero dimensional elimination problems

$$G_1^{(n)} = 0, \ldots, G_n^{(n)} = 0, H^{(n)}$$

given by

$$G_1^{(n)} := X_1^2 - X_1, \ldots, G_n^{(n)} := X_n^2 - X_n$$

and

$$H^{(n)} := \sum_{1 \leq i \leq n} 2^{i-1} X_i \ + \ T \prod_{1 \leq i \leq n} (1 + (U_i - 1) X_i).$$

Let $X_{n+1}, \ldots, X_{3n-1}$ be new indeterminates and let us consider the following polynomials

$$G_{n+1}^{(n)} := X_{n+1} - 2X_2 - X_1, \ldots, G_j^{(n)} := X_j - X_{j-1} - 2^{j-n} X_{j-n+1}, \ n+2 \leq j \leq 2n-1,$$

$$G_{2n}^{(n)} := X_{2n} - U_1 X_1 + X_1 - 1,$$

$$G_k^{(n)} := X_k - U_{k-2n+1} X_{k-1} X_{k-2n+1} + X_{k-1} X_{k-2n+1} - X_{k-1}, \ 2n+1 \leq k \leq 3n-1$$

and

$$L^{(n)} := X_{2n-1} + T X_{3n-1}.$$

One verifies easily that $G_1^{(n)} = 0, \ldots, G_{3n-1}^{(n)} = 0, L^{(n)}$ is another flat family of zero dimensional elimination problems and that both families have the same associated elimination polynomial, namely

$$F^{(n)} := \prod (Y - (j + T \prod_{1 \leq i \leq n} U_i^{[\rho]_i}))$$

Suppose now that there is given an essential division–free procedure $\mathcal{A}$ of our extended computation model which solves algorithmically the general instance of any given flat family of zero–dimensional elimination problems.

Let β_n and β_n^* be two essentially division–free, robust parameterized arithmetic circuits which implement the first and the second flat family of zero dimensional elimination problems we are considering.

Then β_n and β_n^* are necessarily distinct circuits. Therefore $\mathcal{A}_{\text{final}}(\beta_n)$ and $\mathcal{A}_{\text{final}}(\beta_n^*)$ represent two implementations of the elimination polynomial $F^{(u)}$ by essentially division–free, robust parameterized arithmetic circuits.

From Theorem 10 and its proof we are only able to deduce that the circuit $\mathcal{A}_{\text{final}}(\beta_n)$ has non–scalar size at least $\Omega(2^n)$, but we know nothing about the non–scalar size of $\mathcal{A}_{\text{final}}(\beta_n^*)$.

In the past, many attempts to show the non–polynomial character of the elimination of just one existential quantifier block in the arithmetic circuit based elementary language over $\mathbb{C}$, employed the reduction to the proof that a certain sequence of specific polynomials was hard to evaluate (this approach was introduced in [HM93] and became adapted to the BSS model in [SS95]).

The Pochhammer polynomials and the generic permanents discussed below form such sequences.

Let us finish this section with a word about hacking and interactive (zero–knowledge) proofs.

Hackers work in an ad hoc manner and quality attributes are irrelevant for them. We may simulate a hacker and his environment by an *interactive proof system* where the prover, identified with the hacker, communicates with the verifier, i.e., the user of the hacker's program. Thus, in our view, a hacker disposes over unlimited computational power, but his behaviour is deterministic. Only his communication with the user underlies some quantitative restrictions: communication channels are tight. Hacker and user become linked by a protocol of zero–knowledge type which we are going to explain now.

Inputs are natural numbers in *unary* representation. Each natural number represents a mathematical object belonging to a previously fixed abstract data type of polynomials. For example $n \in \mathbb{N}$ may represent the 2^n–th Pochhammer polynomial

$$T^{\underline{2^n}} := \prod_{0 \leq j < 2^n} (T - j)$$

or the n–th generic permanent

$$\mathrm{Perm}_n := \sum_{\tau \in \mathrm{Sym}(n)} X_{1\tau(1)}, \ldots, X_{n\tau(n)},$$

where T and $X_{11}, \ldots, X_{nn}$ are new indeterminates and $\mathrm{Sym}(n)$ denotes the symmetric group operating on n elements.

On input $n \in \mathbb{N}$ the hacker sends to the user a division–free labelled directed acyclic graph Γ_n (i.e., a division–free ordinary arithmetic circuit over $\mathbb{Z}$) of size $n^{O(1)}$ and claims that Γ_n evaluates the polynomial represented by n.

The task of the user is now to check this claim in uniform, bounded probabilistic or non–uniform polynomial time, namely in time $n^{O(1)}$.

In the case of the Pochhammer polynomial and the permanent a suitable protocol exists. This can be formulated as follows.

Proposition 14 *The languages*

$$\begin{aligned}\mathcal{L}_{Poch} \;:=\; & \{(n, (\Gamma_j)_{0\leq j\leq n});\ n \in \mathbb{N},\ \Gamma_j \textit{ is for } 0 \leq j \leq n \\ & \qquad \textit{a division–free labelled directed acyclic graph evaluating } T^{\underline{2^j}}\}\end{aligned}$$

and

$$\mathcal{L}_{Perm} \;:=\; \{(n, \Gamma);\ n \in \mathbb{N}\ , \Gamma \textit{ is a labelled directed acyclic graph evaluating } Perm_n\ \}$$

belong to the complexity class BPP and hence to P/poly (here $n \in \mathbb{N}$ is given in unary representation).

Proof. We show only that $\mathcal{L}_{\mathrm{Poch}}$ belongs to the complexity class *P/poly*. The proof that $\mathcal{L}_{\mathrm{Poch}}$ belongs to BPP follows the same kind of argumentation and will be omitted here. The case of the language $\mathcal{L}_{\mathrm{Perm}}$ can be treated analogously and we shall not do it here (compare [KI04], Section 3).

Let $n \in \mathbb{N}$ and let Γ be a division–free labelled directed acyclic graph with input T and a single output node. Let Γ' be the division–free labelled directed acyclic graph which is given by the following construction:

- choose a labelled acyclic graph μ_n of size $n + O(1)$ with input T and with $T - 2^{2^{n-1}}$ as single final result

- take the union $\overline{\Gamma}$ of the circuits Γ and $\Gamma * \mu_n$ and connect the two output nodes of $\overline{\Gamma}$ by a multiplication node which becomes then the single output node of the resulting circuit Γ'.

From the polynomial identity $T^{\underline{2^n}} = T^{\underline{2^{n-1}}}(T) \ \cdot\ T^{\underline{2^{n-1}}}(T - 2^{2^{n-1}})$ one deduces easily that Γ' computes the polynomial $T^{\underline{2^n}}$ if and only if Γ computes the polynomial $T^{\underline{2^{n-1}}}$.

For $0 \leq j \leq n$ let Γ_j be a division–free labelled directed acyclic graph with input T and a single output node.

Suppose that in the previous construction the circuit Γ is realized by the labelled directed acyclic graph Γ_{n-1}. Then one sees easily that $(n, (\Gamma_j)_{0\leq j\leq n})$ belongs to $\mathcal{L}_{\mathrm{Poch}}$ if and only if the following conditions are satisfied:

(i) the circuit Γ_0 computes the polynomial T,

(ii) the circuits Γ' and Γ_n compute the same polynomial,

(iii) $(n-1, (\Gamma_j)_{0 \leq j \leq n-1})$ belongs to $\mathcal{L}_{\text{Poch}}$.

Therefore, if condition (ii) can be checked in non–uniform polynomial time, the claimed statement follows.

For $0 \leq j \leq n$ let L_j and L be the sizes of the labelled directed acyclic graphs Γ_j and Γ' and observe that $L = 2L_{n-1} + n + O(1)$ holds.

Let P_{n-1} and P be the final results of the circuits Γ_{n-1} and Γ'. From [CGH+03], Corollary 2 we deduce that there exist $m := 4(L+2)^2 + 2$ integers $\gamma_1, \ldots, \gamma_m \in \mathbb{Z}$ of bit length at most $2(L+1)$ such that the condition (ii) above is satisfied if and only if

(iv) $P_{n-1}(\gamma_1) = P(\gamma_1), \ldots, P_{n-1}(\gamma_m) = P(\gamma_m)$

holds.

From [HM] we infer that condition (iv) can be checked by a nondeterministic Turing machine with advise in (non–uniform) time $O(L^3) = O((L_{n-1}+n)^3)$.

Applying this argument recursively, we conclude that membership of $(n, (\Gamma_j)_{0 \leq j \leq n})$ to $\mathcal{L}_{\text{Poch}}$ may be decided in non–uniform time $O(\sum_{0 \leq j \leq n} (L_j + j)^3)$ and therefore in polynomial time in the input size. Hence the language $\mathcal{L}_{\text{Poch}}$ belongs to the complexity class *P/poly*. The proof of the stronger result, namely $\mathcal{L}_{\text{Poch}} \in \text{BPP}$, is similar. ■

Finally we observe that for $n \in \mathbb{N}$ the Pochhammer polynomial $T^{\overline{2^n}}$ is the associated elimination polynomial of the particular problem instance, given by $T := 0$, of the flat family of zero–dimensional elimination problems $G_1^{(n)} = 0, \ldots, G_n^{(n)} = 0, H^{(n)}$, which we considered in Section 4.1.

From the point of view of effective elimination, the sequence of Pochhammer polynomials becomes discussed in [HM93] (see also [SS95]). From the point of view of factoring integers, Pochhammer polynomials are treated in [Lip94].

Let us mention that our approach to effective elimination theory, which led to Theorem 10 and preliminary forms of it, was introduced in [HMPW98] and extended in [GH01] and [CGH+03].

The final outcome of our considerations in Sections 4.1 and 4.5 is the following: neither mathematicians nor software engineers, nor a combination of them will ever produce a practically satisfactory, *generalistic* software for elimination tasks in Algebraic Geometry. This is a job for *hackers* which may find for *particular* elimination problems *specific* efficient solutions.

References

[Ald84] A. Alder. *Grenzrang und Grenzkomplexität aus algebraischer und topologischer Sicht*. PhD thesis, Universität Zürich, Philosophische Fakultät II, 1984.

[Apt81] K. R. Apt. Ten years of Hoare's logic: A survey–part I. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 3 (4) 431–483, 1981.

[Bar68] E. H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22 (103) 565–578, 1968.

[BC97] T. Bloom, J. P. Calvi. A continuity property of multivariate Lagrange interpolation. *Math. Comp.*, 66 (220) 1561–1577, 1997.

[BCS97] P. Bürgisser, M. Clausen, M. A. Shokrollahi. *Algebraic Complexity Theory. Grundlehren der mathematischen Wissenschaften*, 315. Springer Verlag, 1997.

[BCSS98] L. Blum, F. Cucker, M. Shub, S. Smale. *Complexity and Real Computation*. Springer–Verlag, 1998.

[BF91] László Babai, Lance Fortnow. Arithmetization: A new method in structural complexity theory. *Computational Complexity*, 1 41–66, 1991.

[Bla09] R. Blanco. Complexity of Villamayor's algorithm in the non-exceptional monomial case. *International Journal of Mathematics*, 20 (6) 659–678, 2009.

[BSS89] L. Blum, M. Shub, S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 1 (21) 1–45, 1989.

[CGH89] L. Caniglia, A. Galligo, J. Heintz. Some new effectivity bounds in computational geometry. *Applied Algebra, Algebraic Algorithms and Error Correcting Codes. Proc. of the 6th Intern. Conference AAECC, Best Paper Award AAECC-6. Springer LNCS*, 357 131–151, 1989.

[CGH$^+$03] D. Castro, M. Giusti, J. Heintz, G. Matera, L. M. Pardo. The hardness of polynomial equation solving. *Foundations of Computational Mathematics*, 3 (4) 347–420, 2003.

[dBR92] C. de Boor, A. Ron. The least solution for the polynomial interpolation problem. *Math. Z.*, 210 (3) 347–378, 1992.

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

[Edm67] J. Edmonds. Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards*, 71B 241–245, 1967.

[EV00] S. Encinas, O. Villamayor. A course on constructive desingularization and equivariance. *Resolution of singularities: a research textbook in tribute to Oscar Zariski. Progress in Mathematics*, 181 147–227, 2000.

[Ful84] William Fulton. *Intersection Theory.* Number 2 in Ergebnisse der Mathematik und ihre Grenzgebiete. Springer-Verlag, Berlin, 1984. 3. Folge.

[GH01] M. Giusti, J. Heintz. Kronecker's smart, little black boxes. In *Foundations of Computational Mathematics, R. A. DeVore, A. Iserles, E. Süli eds.*, 284 of *London Mathematical Society Lecture Note Series*, 69–104. Cambridge University Press, Cambridge, 2001.

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

[GHKa] M. Giusti, J. Heintz, B. Kuijpers. The evaluation of geometric queries: constraint databases and quantifier elimination. Manuscript University of Buenos Aires (2007).
`http://alpha.uhasselt.be/~lucp1265/publications/sad.pdf`.

[GHKb] R. Grimson, J. Heintz, B. Kuijpers. Evaluating geometric queries with few arithmetic operations. Manuscript University of Buenos Aires (2011).

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

[GHMP97] M. Giusti, J. Heintz, J. E. Morais, L. M. Pardo. Le rôle des structures de données dans les problemes d'élimination. *Comptes Rendus Acad. Sci.*, Serie 1 (325) 1223–1228, 1997.

[GHMS11] N. Giménez, J. Heintz, G. Matera, P. Solernó. Lower complexity bounds for interpolation algorithms. *Journal of Complexity*, 27 151–187, 2011.

[GLS01] M. Giusti, G. Lecerf, B. Salvy. A Gröbner Free Alternative for Polynomial System Solving. *Journal of Complexity*, 17 154–211, 2001.

[Har92] J. Harris. *Algebraic geometry: a first course.* Springer Verlag, 2. edition, 1992.

[Hei83] J. Heintz. Definability and fast quantifier elimination in algebraically closed fields. *Theor. Comput. Sci.*, 24 239–277, 1983.

[Hei89] J. Heintz. On the computational complexity of polynomials and bilinear mappings. A survey. *Proceedings 5th International Symposium on Applied Algebra, Algebraic Algorithms and Error Correcting Codes Springer LNCS*, 356 269–300, 1989.

[HK04] J. Heintz, B. Kuijpers. Constraint databases, data structures and efficient query evaluation. *Constraint Databases. First International Symposium Springer LNCS*, 3074 1–24, 2004.

[HM] K. Hägele, J. L. Montaña. Polynomial random test for the equivalence of integers given by arithmetic circuits. Preprint 4/97, Departamento de Matemática, Estadística y Computación, Universidad de Cantabria (1997).

[HM93] J. Heintz, J. Morgenstern. On the intrinsic complexity of elimination theory. *Journal of Complexity*, 9 471–498, 1993.

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

[HS80] J. Heintz, C.P. Schnorr. Testing polynomials which are easy to compute. *International Symposium on Logic and Algorithmic. Monogr. Enseig. Math. 30, 237–254, 1982 and 12th Annual Symposium ACM on Theory of Computing (STOC' 80) ACM Press*, 262–272, 1980.

[Ive73] B. Iversen. *Generic Local Structure of the Morphisms in Commutative Algebra.* Springer-Verlag, Berlin, 1973.

[KI04] V. Kabanets, R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 1-2 (13) 1–46, 2004.

[KP96] T. Krick, L. M. Pardo. A computational method for diophantine approximation. *Algorithms in Algebraic Geometry and Applications. Proceedings of MEGA'94. Progress in Mathematics*, 143 193–254, 1996.

[Kun85] E. Kunz. *Introduction to commutative algebra and algebraic geometry.* Birkhäuser, Boston, 1985.

[Lan93] S. Lang. *Algebra.* Addison-Wesley, Massachusetts, 1993.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39 859–868, 1992.

[Lic90] T. M. Lickteig. On semialgebraic decision complexity. Habilitationsschrift, Universität Tübingen TR-90-052, Int. Comp. Sc. Inst., Berkeley, 1990.

[Lip94] R. J. Lipton. Straight–line complexity and integer factorization. *Algorithmic number theory. Springer LNCS*, 877 71–79, 1994.

[Mey00] B. Meyer. *Object-Oriented Software Construction.* Prentice-Hall, 2. edition, 2000.

[Mor03] T. Mora. *SPES I: The Kronecker-Duval Philosophy.* Cambridge University Press, 2003.

[Mor05] T. Mora. *SPES II: Macaulay's Paradigm and Groebner Technology.* Cambridge University Press, 2005.

[Mum88] D. Mumford. *The red book of varieties and schemes*, 1358. Springer, Berlin Heidelberg, New York, 1. edition, 1988.

[Olv06] P. Olver. On multivariate interpolation. *Stud. Appl. Math.*, 116 (2) 201–240, 2006.

[PS73] M. S. Paterson, L. J. Stockmeyer. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing*, 2 60–66, 1973.

[Sax09] N. Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 90 49–79, 2009.

[Sha92] A. Shamir. IP=PSPACE. *J. ACM*, 39 869–877, 1992.

[Sha94] I. R. Shafarevich. *Basic algebraic geometry: Varieties in projective space.* Springer, Berlin Heidelberg, New York, 1994.

[Shp10] A. Shpilka. Arithmetic circuits: a survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5 (3-4) 207–388, 2010.

[SS95] M. Shub, S. Smale. On the intractability of Hilbert's Nullstellensatz and an algebraic version of "NP≠P?". *Duke Math. J.*, 81 47–54, 1995.

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

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

[ZS60] O. Zariski, P. Samuel. *Commutative algebra II*, 39. Springer, New York, 1960.