Directed evaluation

Joris van der Hoeven, Grégoire Lecerf

Directed evaluation*

JORIS VAN DER HOEVEN[a], GRÉGOIRE LECERF[b]

CNRS, École polytechnique, Institut Polytechnique de Paris
Laboratoire d'informatique de l'École polytechnique (LIX, UMR 7161)
1, rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing, CS35003
91120 Palaiseau, France

a. Email: vdhoeven@lix.polytechnique.fr
b. Email: lecerf@lix.polytechnique.fr

Final version of October 2020

Let $\mathbb{K}$ be a fixed effective field. The most straightforward approach to compute with an element in the algebraic closure of $\mathbb{K}$ is to compute modulo its minimal polynomial. The determination of a minimal polynomial from an arbitrary annihilator requires an algorithm for polynomial factorization over $\mathbb{K}$. Unfortunately, such algorithms do not exist over generic effective fields. They do exist over fields that are explicitly generated over their prime sub-field, but they are often expensive. The dynamic evaluation paradigm, introduced by Duval and collaborators in the eighties, offers an alternative algorithmic solution for computations in the algebraic closure of $\mathbb{K}$. This approach does not require an algorithm for polynomial factorization, but it still suffers from a non-trivial overhead due to suboptimal recomputations. For the first time, we design another paradigm, called directed evaluation, which combines the conceptual advantages of dynamic evaluation with a good worst case complexity bound.

KEYWORDS: complexity, algorithm, computer algebra, algebraic extension, algebraic tower, triangular set, dynamic evaluation, directed evaluation, accelerated tower.

1. INTRODUCTION

Let $\mathbb{K}$ be an effective field. Here, *effective* means that elements of $\mathbb{K}$ can be represented by concrete data structures and that algorithms for the field operations are at our disposal, including a zero-test. Effective rings are defined in a similar way.

1.1. Statement of the problems

The aim of the present paper is the fast computation with algebraic numbers. We start by recalling various known strategies, including the most prominent one: dynamic evaluation. Our first main problem will be to develop a more efficient variant of dynamic evaluation. We next recall how to represent algebraic numbers using towers of algebraic extensions. Our second main problem is to develop efficient algorithms for computations in such towers.

*. This paper is part of a project that has received funding from the French "Agence de l'Innovation de Défense".

Dynamic evaluation Let μ be a monic separable polynomial of degree d in $\mathbb{K}[x]$, not necessarily irreducible, and let ζ represent a root of μ in the algebraic closure $\bar{\mathbb{K}}$ of $\mathbb{K}$. Given an algorithm T that runs over a generic effective field, consider the question of running T efficiently over $\mathbb{K}[\zeta]$.

A natural first approach begins with the computation of the defining polynomial of ζ over $\mathbb{K}$: this essentially corresponds to factoring μ into irreducible polynomials. Unfortunately this task is not feasible over a generic field [19, 20]. Of course, when $\mathbb{K}$ is explicitly finitely generated over its prime sub-field, such factorizations can be computed. However, no general efficient algorithms (i.e. running in softly linear time) are currently known for this task; see [21].

Another approach consists in computing in $\mathbb{K}[\alpha]$ while regarding α as a parameter constrained by $\mu(\alpha)=0$. So any element a in $\mathbb{K}[\alpha]$ is represented by a polynomial A in $\mathbb{K}[x]$ of degree $<d$, such that $a=A(\alpha)$. With this representation, additions, subtractions and products can easily be performed in $\mathbb{K}[x]/(\mu(x))$, with α seen as the class of x. On the other hand, testing whether $A(\alpha)$ is zero or not requires us to distinguish the following cases:

- If A is the zero polynomial, then $A(\zeta)$ is zero for any root ζ of μ, so the result of the test is "true";
- If the resultant $\mathrm{Res}(A,\mu)$ is non-zero, then $A(\zeta)$ is non-zero for any root ζ of μ, so the result of the test is "false";
- Otherwise, $g:=\gcd(A,\mu)$ is not a constant polynomial and has degree $<d$. A proper decomposition of μ is thus discovered: $\mu=g\,(\mu/g)$. We *split* the current computation, into the two following ones:
  - In the first branch, α is subjected to the constraint $g(\alpha)=0$; in this branch, the result of the zero-test $A(\alpha)=0$ becomes "true".
  - In the second branch, α is subjected to the constraint $(\mu/g)(\alpha)=0$ and the result of the zero-test $A(\alpha)=0$ becomes "false".

If the inverse of $A(\alpha)$ is requested, then we first test if $A(\alpha)$ is zero or not. If A is identically zero (in the resulting branch), then an exception is raised; if $\mathrm{Res}(A,\mu)$ is non-zero, then the inverse can be computed from the Bézout relation for A and μ, which can itself be computed using the extended gcd algorithm.

This manner of evaluating a program is called *dynamic evaluation*. It finishes after a necessarily finite number of splittings. At the end of all the computations, we obtain a factorization of $\mu=\mu_1\cdots\mu_s$, not necessarily into irreducible polynomials, along with one output value for each of the s cases when $\mu_i(\alpha)=0$ for $i=1,\ldots,s$. In other words, instead of working with a specific root of μ, the program is evaluated for a generic root of μ. For this reason, α is sometimes called an *algebraic parameter*: the program is executed as if $\mathbb{K}[\alpha]$ were a field, and cases are distinguished only when inconsistencies actually occur during the execution. After a zero test $a=A(\alpha)=0$ or an inversion a^{-1} that gives rise to a splitting, the element a is enforced to be identically zero or invertible in each branch. Notice that the separability assumption on μ is important, since it ensures that g and μ/g are coprime in the above decomposition $\mu=g\,(\mu/g)$.

Example 1.1. Let α be a parameter with $\mu(\alpha)=0$ for $\mu(x):=x\,(x-1)\,(x-2)\in\mathbb{Q}[x]$, and consider the computation of the gcd of

$$f(y):=y^2-(\alpha+\alpha^2)\,y+3\,\alpha^2-2\,\alpha,\ \text{and}\ g(y):=(\alpha-1)\,y^2-3\,(\alpha-1)\,y+2\,(\alpha-1).$$

Using dynamic evaluation, this gives rise to three branches, and the result

$$\gcd(f,g) = \begin{cases} 1 & \text{if } \alpha = 0 \\ 0 & \text{if } \alpha - 1 = 0 \\ y - 2 & \text{if } \alpha - 2 = 0. \end{cases}$$

Dynamic evaluation is very attractive from a conceptual point of view, but rather tricky to implement: splittings correspond to non-deterministic control structures that are typically implemented using continuations [41]. Dynamic evaluation is also difficult to analyze from the complexity perspective. In particular, it is hard to determine the amount of computation that can be shared between several branches. At any rate, the worst case complexity of dynamic evaluation is much higher than the number of steps of the program multiplied by the cost of operations in $\mathbb{K}[x]/(\mu(x))$.

The first aim of this paper is to design an alternative, so-called *directed* evaluation strategy, with the same abstract specification, but with a quasi-linear computational complexity in d. The key idea is to run the entire program, while systematically opting for the "principal" branch in case of splittings. This principal branch is the one for which the defining polynomial of α is maximal. At every splitting, we also store the defining polynomial for the other "residual" branch. Once the computation for the principal branch has been completed, we recursively apply the same algorithm for each residual branch, on the reduced input data modulo the corresponding defining polynomial. The required reductions are computed simultaneously for all residual branches, using a fast divide-and-conquer algorithm for multi-modular reduction.

Algebraic towers We have outlined above how to compute with a single algebraic parameter α using the dynamic evaluation strategy. More generally, one may wish to compute with a finite number of parameters $\alpha_1, \ldots, \alpha_t$ that are introduced successively as follows: α_1 is subjected to the constraint $\mu_1(\alpha_1) = 0$ for some monic polynomial $\mu_1 \in \mathbb{K}[x_1]$, then α_2 is subjected to $\mu_2(\alpha_2) = 0$ with μ_2 monic in $\mathbb{K}[\alpha_1][x_2]$, then α_3 is subjected to $\mu_3(\alpha_3) = 0$ with μ_3 monic in $\mathbb{K}[\alpha_1, \alpha_2][x_3]$, and so on. The second aim of the present paper is the design of fast algorithms for computing dynamically in $\mathbb{K}[\alpha_1, \ldots, \alpha_t]$ as if it were a field. The main difficulties arise when t is allowed to be arbitrarily large.

The parameters $\alpha_1, \ldots, \alpha_t$ naturally induce a *tower* of effective algebraic extensions

$$\mathbb{K} = \mathbb{A}_0 \subseteq \mathbb{A}_1 \subseteq \mathbb{A}_2 \subseteq \cdots \subseteq \mathbb{A}_t,$$

where

$$\begin{aligned} \mathbb{A}_0 &:= \mathbb{K} \\ \mathbb{A}_i &:= \mathbb{A}_{i-1}[x_i]/(\mu_i(x_i)), \qquad \text{for } i = 1, \ldots, t. \end{aligned}$$

The parameter α_i corresponds to the class of x_i in $\mathbb{A}_i$ for $i = 1, \ldots, t$. A tower of this type is written $(\mathbb{A}_i)_{i \leqslant t}$ and we call t its *height*. Throughout this paper, we write $d_i := \deg \mu_i$ and $d := d_1 \cdots d_t$. The μ_i are called the *defining polynomials* of the tower, and d its *degree*. Elements in $\mathbb{A}_i$ are naturally represented by univariate polynomials in α_i over $\mathbb{A}_{i-1}$ of degree $< d_i$.

In order to reduce computational costs, it will be convenient to assume that the tower is *explicitly separable* in the sense that $\gcd(\mu_i', \mu_i) = 1$ and that we have explicit Bézout cofactors $\theta_i, \xi_i \in \mathbb{A}_{i-1}[x_i]$ with $\theta_i \mu_i' + \xi_i \mu_i = 1$ for $i = 1, \ldots, t$. In particular, each $\bar{\mathbb{K}} \otimes \mathbb{A}_i$ is reduced. This means that $\bar{\mathbb{K}} \otimes \mathbb{A}_t$ contains no non-zero nilpotent elements or, equivalently, that $\mathbb{A}_i$ is a product of separable field extensions of $\mathbb{K}$.

Towers of algebraic extensions are related to "triangular sets". In fact, we may see the preimage of μ_i over $\mathbb{K}$ as a multivariate polynomial T_i in $\mathbb{K}[x_1, \ldots, x_i]$ such that T_i has degree d_i in x_i, degrees $<d_j$ in x_j for $j=1,\ldots,i-1$, and $\mu_i(x_i) = T_i(\alpha_1, \ldots, \alpha_{i-1}, x_i)$. The sequence $(T_i)_{i\leqslant t}$ then forms a special regular case of a *triangular set*. The separability of μ_i translates into the requirement that $T_i(\zeta_1, \ldots, \zeta_{i-1}, x_i)$ is separable for all roots ζ_j of $T_j(\zeta_1, \ldots, \zeta_{j-1}, x_j)$ for $j=1,\ldots,i-1$.

The second main aim of this paper is to generalize the directed evaluation strategy from the univariate case, while ensuring that the complexity remains bounded by the number of steps in our algorithm multiplied by $d^{1+o(1)}$ (which corresponds roughly speaking to the cost of arithmetic operations in the tower [29]). Intuitively speaking, we use the univariate strategy in a recursive fashion, but the analysis becomes far more technical due to the fact that splittings can occur at any level of the tower. This difficulty will be countered through the development of suitable data structures.

Further notations Before we discuss complexity issues of dynamic evaluation and tower computations in more detail, it is useful to introduce various standard notations. We often use the *soft-Oh* notation: $f(n) \in \tilde{O}(g(n))$ means that $f(n) = g(n)\log^{O(1)}(g(n)+3)$; see [21, chapter 25, section 7] for technical details. The notation $f = \Omega(g)$ means that there exist $c>0$ such that $f(n) \geqslant c\,g(n)$ for n sufficiently large. The least integer larger or equal to x is written $\lceil x \rceil$; the largest one smaller or equal to x is written $\lfloor x \rfloor$.

Given an effective ring $\mathbb{A}$, we use $\mathsf{M}_{\mathbb{A}}(d)$ as a notation for the cost of multiplying two polynomials of degree $<d$, in terms of the number of operations in $\mathbb{A}$. It is known [6] that $\mathsf{M}_{\mathbb{A}}(d) = O(d\log d \log\log d)$. We will also denote

$$\mathbb{A}[x]_{<d} := \{P \in \mathbb{A}[x] : \deg P < d\}.$$

In particular, given $\mathbb{B} = \mathbb{A}[x]/(\mu(x))$ for some monic polynomial $\mu \in \mathbb{A}[x]$ of degree d, elements in $\mathbb{B}$ may be represented as classes of polynomials in $\mathbb{A}[x]_{<d}$ modulo μ, additions in $\mathbb{B}$ require d additions in $\mathbb{A}$, whereas multiplications in $\mathbb{B}$ can be done using $O(\mathsf{M}_{\mathbb{A}}(d))$ ring operations in $\mathbb{A}$.

We let $\omega \leqslant 3$ denote an exponent such that two $n\times n$ matrices over a commutative ring $\mathbb{A}$ can be multiplied with $O(n^\omega)$ operations in $\mathbb{A}$. The constant $\varpi \leqslant 2$ denotes another exponent such that a rectangular $n \times \sqrt{n}$ matrix over a commutative ring $\mathbb{A}$ can be multiplied with a square $\sqrt{n}\times\sqrt{n}$ matrix with $O(n^\varpi)$ operations in $\mathbb{A}$. Le Gall has shown in [34] that one may take $\omega < 2.373$; Huang and Pan have shown in [31] that one may take $\varpi < 1.667$. Throughout this paper, in order to simplify complexity analyses, we assume that $\omega > 2$ and $\varpi > {}^3\!/_2$.

1.2. Previous work

Recall that an element $\beta \in \mathbb{A}_t$ is said to be *primitive* if $\mathbb{A}_t \cong \mathbb{K}[\beta]$. If $\bar{\mathbb{K}} \otimes \mathbb{A}_t$ is a reduced $\bar{\mathbb{K}}$-algebra, then it is well known that a sufficiently generic $\mathbb{K}$-linear combination of $\alpha_1, \ldots, \alpha_t$ is a primitive element of $\mathbb{A}_t$. It may thus seem odd, at first sight, to manipulate towers $(\mathbb{A}_i)_{i\leqslant t}$ of height $t \geqslant 2$ whenever an isomorphic algebra $\mathbb{K}[\beta]$ exists. The problem is that both the computations of primitive elements and conversions between representations are usually expensive.

Over a general field $\mathbb{K}$, no efficient algorithm is currently known for finding a primitive element together with its minimal polynomial ν, and polynomials $\varphi_i \in \mathbb{K}[x]_{<d}$ such that $\alpha_i = \varphi_i(\beta)$ for $i=1,\ldots,t$. It is known that this problem can efficiently be reduced to

a multivariate version of the problem of modular composition using a randomized Las Vegas algorithm. The corresponding conversions between tower and primitive element representations can also be done using modular composition. All this is a consequence of the transposition principle and Le Verrier's method; we refer to [29, 30, 33, 38, 39] for more details. Unfortunately, no softly linear algorithm is known for modular composition over a generic field, although efficient algorithms have recently been designed for specific cases [9, 27, 28]. If $\mathbb{K}$ is a finite field, then Kedlaya–Umans and followers have established almost optimal theoretical bit complexity bounds [30, 33, 38].

Without primitive elements, using a naive induction on t, the multiplication in $\mathbb{A}_t$ can be performed in time $O(d\,(K \log d \log\log d)^t)$ for some constant $K > 1$. It is well known that, for a sufficiently large constant $C > 1$, such products can actually be carried out in time $O(C^t d \log d \log\log d)$ using Kronecker substitution; see for instance [21, chapter 8]. If many of the individual degrees d_i are small, then t can become as large as $\log d / \log 2$, which leads to an overall complexity bound of the form $d^{1+\log C/\log 2+o(1)}$. Unfortunately, this bound is far from linear if C is much larger than 1. Lebreton has shown that one may take $C = 3$; see [35] and [29, Proposition 2.4].

Substantial improvements for the value of C seem difficult to achieve, and it might not even be possible to reach values that are arbitrarily close to 1. An alternative approach for efficient computations in a given tower is to produce an equivalent, so-called *accelerated* tower of small height, such that conversions between both towers are reasonably cheap. This approach was first proposed in [29]: when the μ_i are all irreducible, we have shown how to multiply elements in $\mathbb{A}_t$ with $O(d^{1+\varepsilon})$ operations in $\mathbb{K}$, for any fixed real $\varepsilon > 0$.

Dynamic evaluation has been developed by Della Dora, Dicrescenzo and Duval [10, 13, 14, 15] as a way to compute with algebraic parameters without irreducible factorizations. The approach is sometimes called the "D5 principle", after the initials of the authors of [10]. One may regard dynamic evaluation as a computer algebra counterpart of the concept of non-deterministic evaluation from theoretical computer science. The strategy of dynamic evaluation has been extended to transcendental parameters [17, 22], real algebraic numbers in [16], and more general algebraic structures [25, chapter 8].

Several implementations have been proposed for parameters that are constrained by triangular sets. Early implementations simply handled splittings by redoing the entire computation under stricter constraints [13]. Unnecessary recomputations can be avoided through the use of high-level control structures such as continuations [4]. Unfortunately, efficient implementations of such control structures are rarely available for common programming languages. In this paper, we mostly leave implementation issues aside and focus on the complexity of computations with parameters.

In his PhD thesis, Dellière has investigated the relationship between dynamic evaluation and decompositions of constructible sets into triangular sets [11, 12]: the central operation is the computation of gcds with coefficients in products of fields, such as $\mathbb{A}_t$. Let us further mention that dynamic evaluation has influenced several polynomial system solvers relying on triangular sets [3], and is now involved in various other algorithms in computer algebra; see for instance [7, 26, 36].

Dedicated algorithms over dynamic fields such as $\mathbb{A}_t$ have been proposed by Dahan *et al.* [8]. Without appealing to the general dynamic evaluation paradigm, they designed efficient algorithms for quasi-inversion in $\mathbb{A}_t$, as well as gcd computations and coprime factorization in $\mathbb{A}_t[x]$. Recomputations induced by splittings are handled using fast *ad hoc* techniques.

In the worst case, it is known that dynamic evaluation over $\mathbb{A}_t$ suffers from an overhead of the order $d := \dim_{\mathbb{K}} \mathbb{A}_t$: see Examples 4.4 and 4.6. To our knowledge, this paper is first to propose a general strategy for removing this overhead.

1.3. Our contributions

The main contribution of the present paper is a new evaluation paradigm to compute with algebraic parameters in an asymptotically fast manner. In section 2, we recall the concept of computation trees as a formalization for the cost of algebraic algorithms. In section 3, we introduce the concepts of unpermissive, panoramic, and directed evaluation. This provides us with precise terminology for analyzing subtle differences between the complexities of various evaluation strategies.

Informally speaking, the unpermissive evaluation of a computation tree simply aborts the usual evaluation whenever a zero-divisor needs to be inverted or tested to zero. If a zero-divisor is discovered in this way, then a proper factorization $\mu_i = \hat{\mu}_i \breve{\mu}_i$ of one of the defining polynomials can be deduced. This leads to a decomposition of $\mathbb{A}_t$ into the direct sum of two proper subalgebras, defined by the towers $\mathbb{A}_0 \subseteq \cdots \subseteq \mathbb{A}_{i-1} \subseteq \mathbb{A}_{i-1}[x_i]/(\hat{\mu}_i(x_i)) \subseteq \cdots$ and $\mathbb{A}_0 \subseteq \cdots \subseteq \mathbb{A}_{i-1} \subseteq \mathbb{A}_{i-1}[x_i]/(\breve{\mu}_i(x_i)) \subseteq \cdots$.

A panoramic evaluation of a computation tree returns all possible results when considering $\alpha_1, \ldots, \alpha_t$ as algebraic parameters, and thereby constitutes our main objective. The unpermissive evaluation model gives rise to the following naive panoramic evaluation strategy: whenever the unpermissive evaluation produces a proper splitting of $\mathbb{A}_t$, we restart the evaluation over each of the two subalgebras of $\mathbb{A}_t$. The problem with this rather naive approach is that the evaluation may need to be restarted as many as $d = \dim_{\mathbb{K}} \mathbb{A}_t$ times, which turns out to be suboptimal from a complexity point of view: see Example 4.4.

The directed evaluation is meant to remedy this situation by ensuring a sharp decrease of d every time that we need to restart the evaluation. More precisely, we will show that the reevaluation depth remains bounded by $O(\log d)$.

Section 4.2 contains our main complexity result in the case of a single parameter, i.e. $t = 1$. We prove a softly linear complexity bound for panoramic evaluation and provide a detailed comparison with various implementations of dynamic evaluation.

In section 5, we turn to the case of an arbitrary number of parameters $t \geqslant 2$. With respect to the univariate case, the main difference is that $\mathbb{A}_{t-1}$ is not necessarily a field, although we still conduct our computation as if it were (e.g. during gcds computations in $\mathbb{A}_{t-1}[x]$). Consequently, zero-tests and inversions of elements in $\mathbb{A}_t \setminus \mathbb{A}_{t-1}$ may lead to additional case distinctions according to the possible values of $\alpha_1, \ldots, \alpha_{t-1}$. Despite these technical complications, we again prove a quasi-linear complexity bound for the cost of panoramic evaluation, under the assumption that t is fixed (this means that t is not allowed to grow with d).

For arbitrary, not necessarily bounded heights t, the complexity analysis becomes tedious. We need to revisit the construction of accelerated towers of fields, introduced in [29], in the context of directed evaluation. The key idea is as follows: before the directed evaluation of a given computation tree, we perform the directed computation of an accelerated version of the current tower, so the computation tree can subsequently benefit from accelerated arithmetic. Overall, in section 7, we achieve an overhead of the form d^{ϵ} for the panoramic evaluation, where ϵ represents any fixed positive real number.

Our techniques are more general than those of [8]: they turn out to be applicable in all contexts that involve dynamic evaluation with algebraic numbers. In addition, thanks to our accelerated tower arithmetic, we improve the complexity bounds for the specific tasks studied in [8], whenever t is allowed to be arbitrarily large.

Section 7.4 contains two important additional examples. We first show that products in a separable tower can be done in time $O(d^{1+o(1)})$, which extends [29]. We next present a less straightforward application of fast panoramic evaluation to matrix multiplication with entries in $\mathbb{A}_t$.

2. PREREQUISITES

This section gathers known definitions and results about elementary operations in computer algebra, to be used in this paper.

2.1. Computation trees

Let us recall the notion of a computation tree; we essentially follow the presentation from [5, chapter 4, section 4]. In the present paper, all computation trees manipulate data in $\mathbb{K}$-algebras over an effective field $\mathbb{K}$, and only the following operations are allowed:

- The binary arithmetic operations $+, -, \times$ in the algebra.
- The unary operation invert of inversion, which is partially defined in the algebra.
- The unary zero-test, written zero?.
- For each constant $c \in \mathbb{K}$, the nullary constant function $() \mapsto c$ and the unary function $x \mapsto c\,x$ of scalar multiplication by c. We denote the corresponding sets of constant functions and scalar multiplications by $\mathbb{K}_0$ and $\mathbb{K}_1$.

We write $\mathcal{A} := \mathbb{K}_0 \cup \mathbb{K}_1 \cup \{+, -, \times, \mathsf{invert}\}$ for the set of all arithmetic operations.

Definition Consider a binary tree whose set of nodes is a disjoint union $\mathcal{I} \amalg \mathcal{C} \amalg \mathcal{B} \amalg \mathcal{O}$ of the sets $\mathcal{I}$, $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{O}$ of *input nodes*, *computation nodes*, *branching nodes* and *output nodes*. We assume that the input nodes form an initial segment of unary nodes starting at the root, that the output nodes are leafs (of arity zero), that computation nodes admit arity one, and that branching nodes admit arity two. A *computation tree* T over $\mathbb{K}$ is such a binary tree, together with an *instruction function* that

- assigns an instruction of the form
$$(\mathsf{op}; u_1, \ldots, u_m)$$
to every computation node v, where $\mathsf{op} \in \mathcal{A}$ has arity m, and $u_1, \ldots, u_m$ are nodes in $\mathcal{I} \cup \mathcal{C}$ that are *predecessors* of v (i.e. nodes in the path ascending from v to the root of the tree);
- assigns an instruction of form
$$(\mathsf{zero?}; u),$$
to every branching node v, where u is a predecessor of v in $\mathcal{I} \cup \mathcal{C}$;
- assigns a *return value* $(u_1, \ldots, u_m)$ to every output node v, where $u_1, \ldots, u_m$ are predecessors of v in $\mathcal{I} \cup \mathcal{C}$, and $m = m(v)$ may depend on v.

Evaluation Let T be a computation tree as above with $|\mathcal{I}|$ input nodes. Let $\mathbb{A}$ be a $\mathbb{K}$-algebra and let us show how T gives rise to an evaluation function
$$\begin{aligned} \mathcal{E}(T;.)\colon \mathbb{A}^{|\mathcal{I}|} &\to \{\mathsf{undefined}\} \cup \bigcup_{v \in \mathcal{O}} \mathbb{A}^{m(v)}, \\ a &\mapsto \mathcal{E}(T;a). \end{aligned}$$

The value undefined is a new symbol that is returned whenever an arithmetic operation cannot be executed. In the present framework this can only happen for inversions.

Given an input value $a = (a_1, \ldots, a_{|\mathcal{I}|}) \in \mathbb{A}^{|\mathcal{I}|}$, the evaluation of T at a proceeds by constructing a path $P(T;a)$ from the root of the tree, along with the function

$$\mathcal{E}\colon P(T;a) \to \mathbb{A} \cup \{\mathsf{false}, \mathsf{true}, \mathsf{undefined}\} \cup \bigcup_{v \in O} \mathbb{A}^{m(v)}$$

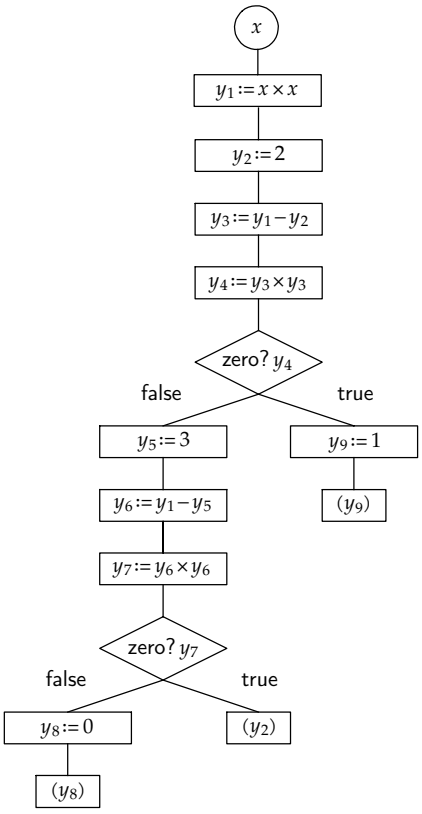
that associates a value to each node of the path, as follows:

- The path $P(T;a)$ begins with the input nodes $u_1, \ldots, u_{|\mathcal{I}|}$ of $\mathcal{I}$, and we set $\mathcal{E}(u_i;a) := a_i$ for $i = 1, \ldots, |\mathcal{I}|$. The next node of the path is set to the successor of $u_{|\mathcal{I}|}$.
- If the current node of the path is a computation node v of the form $(\mathsf{op}; u_1, \ldots, u_m)$ then we set $\mathcal{E}(v;a) := \mathsf{op}(\mathcal{E}(u_1;a), \ldots, \mathcal{E}(u_m;a))$. If this calculation is well defined, then the next node of the path is set to the successor of v. Otherwise, the evaluation path ends at v, we say that the computation tree is *undefined* at a, and we set $\mathcal{E}(T;a) := \mathcal{E}(v;a) :=$ undefined.
- If the current node of the path is a branching node v of the form $(\mathsf{zero?}; u)$, then we set $\mathcal{E}(v;a) := \mathsf{zero?}(\mathcal{E}(u;a))$. If $\mathcal{E}(v;a)$ is false then the next node of the path is set to the left successor of v, otherwise the next node is the right successor of v.
- If the current node of the path is an output node v with return value $(u_1, \ldots, u_m)$, then we set $\mathcal{E}(v;a) := (\mathcal{E}(u_1;a), \ldots, \mathcal{E}(u_m;a))$. The path ends at v and we set $\mathcal{E}(T;a) := \mathcal{E}(v;a)$.

Example 2.1. This example illustrates the definition of a computation tree with $\mathbb{K} = \mathbb{Q}$ and a single input node. Let p_i represent the i-th prime number, let κ be a positive integer, and let T be the computation tree that takes x as input and performs the following computation:

```
for i from 1 to κ do
   if (x^2 − p_i)^κ = 0 then return i
return 0
```

The computation tree for $\kappa = 2$ is shown on the right-hand side. The input node is a circle, computation nodes are long rectangles, output nodes are shorter rectangles, and branching nodes are diamond-shaped.



If $\kappa = 2$, then T evaluates to 1 at a when $a^2 = 2$, to 2 when $a^2 = 3$, and to 0 in all other cases.

We notice that

$$(x^2 - p_i)^\kappa = 0 \Leftrightarrow x^2 = p_i,$$

which makes the example slightly artificial.

Cost functions One may associate cost functions to a computation tree T. Given a path of length l with leaf $v \in \mathcal{O}$ in T, we define $l + m(v)$ to be its cost. The maximal cost of a path is called the *operational cost* of T and we denote it by τ_{op}. We may use τ_{op} as a bound for the number of operations in $\mathbb{A}$ that are required for the evaluation of T over $\mathbb{A}$.

Since additions, multiplications and inversions typically admit different costs, it is convenient to introduce the following more detailed quantities:

- τ_{in} stands for the number of input nodes, i.e. $\tau_{\text{in}} := |\mathcal{I}|$.
- τ_{out} stands for the maximal arity of a return value, i.e. $\tau_{\text{out}} := \max_{v \in \mathcal{O}} m(v)$.
- τ_{add} stands for the maximal number of operations in $\mathbb{K}_0 \cup \mathbb{K}_1 \cup \{+, -\}$ for a path in T.
- τ_{mul} stands for the maximal number of multiplications for a path in T.
- τ_{div} stands for the maximal number of inversions or zero-tests for a path in T.

We will call $(\tau_{\text{in}}, \tau_{\text{out}}, \tau_{\text{add}}, \tau_{\text{mul}}, \tau_{\text{div}})$ the *detailed cost* of T and notice that $\tau_{\text{op}} \leqslant \tau_{\text{in}} + \tau_{\text{out}} + \tau_{\text{add}} + \tau_{\text{mul}} + \tau_{\text{div}}$. Taking $\kappa = 2$ in Example 2.1, we have $\tau_{\text{in}} = 1$, $\tau_{\text{out}} = 1$, $\tau_{\text{add}} = 5$, $\tau_{\text{mul}} = 3$, $\tau_{\text{div}} = 2$.

Usually, we are really interested in the maximal number $\tau_{\mathbb{A}}$ of scalar operations in $\mathbb{K}$ that are needed to evaluate T over $\mathbb{A}$. If $\mathbb{K}$ and $\mathbb{A}$ are clear from the context, this is simply called the *cost* of T or the *time* needed to evaluate T. It will be useful to introduce the following additional quantities:

- $\mathsf{s}_{\mathbb{A}}$ stands for the maximal number of elements in $\mathbb{K}$ that are needed to represent an element in $\mathbb{A}$. Throughout this paper, we assume that operations in $\mathbb{K}_0 \cup \mathbb{K}_1 \cup \{+, -\}$ can all be computed using $O(\mathsf{s}_{\mathbb{A}})$ operations in $\mathbb{K}$.
- $\mathsf{m}_{\mathbb{A}}$ stands for the number of operations in $\mathbb{K}$ that are needed to multiply two elements in $\mathbb{A}$.
- $\mathsf{d}_{\mathbb{A}}$ stands for the maximal number of operations in $\mathbb{K}$ that are needed to perform a zero-test or inversion in $\mathbb{A}$.

We clearly have

$$\tau_{\mathbb{A}} \leqslant O((\tau_{\text{in}} + \tau_{\text{out}} + \tau_{\text{add}})\, \mathsf{s}_{\mathbb{A}}) + \tau_{\text{mul}}\, \mathsf{m}_{\mathbb{A}} + \tau_{\text{div}}\, \mathsf{d}_{\mathbb{A}}.$$

Remark 2.2. In a natural programming language, negations, divisions and equality tests are allowed, but for the sake of the presentation, we assume that

- negations $-a$ are implemented as $0 - a$;
- divisions a / b are implemented as $a \times \mathsf{invert}(b)$;
- equality tests $a = b$ are implemented as $\mathsf{zero?}(a - b)$.

These restrictions are harmless, in the sense that they only affect the constants hidden in the "O" of complexity estimates.

Remark 2.3. For actual implementations of zero-tests that we will encounter in this paper, it is usually possible to compute the actual inverses of non-zero elements with little extra cost. We will allow ourselves to store such inverses "for future use". In gcd computations, this helps us to keep the number of zero tests and inversions sufficiently low. An alternative, more rigorous, but less pedagogic approach would be to systematically work with a hybrid instruction that both performs a zero-test and an inversion in the case when the zero-test fails.

Remark 2.4. The "BSS model" [2, 18] provides an alternative approach for computations over an abstract field $\mathbb{K}$. Roughly speaking, it is a natural extension of the Turing machine model for which the tapes are allowed to contain elements in $\mathbb{K}$. The BSS model is more powerful than the computation tree model in the sense that it allows for loops, subroutines, lookup tables, etc. Nevertheless, an arbitrary program in the BSS model that admits a finite number of execution flows may be emulated by a computation tree. This amounts to "unrolling" all possible executions as in Example 2.1 above.

Remark 2.5. We will also use the notion of a *straight-line program*. In the present context it is convenient to define a straight-line program as a computation tree without branching nodes or inversions, so it encodes a polynomial expression. This definition is a bit different but equivalent to the standard one [5, chapter 4, section 1].

2.2. Univariate polynomials

A polynomial in $\mathbb{A}[x]$ of degree $<d$ is represented by the vector of its d coefficients from degree 0 to $d-1$. Polynomial additions (resp. subtractions) in $\mathbb{A}[x]_{<d}$ take at most d additions (resp. subtractions) in $\mathbb{A}$. Determining the degree of a polynomial in $\mathbb{A}[x]_{<d}$ requires at most d zero-tests.

Product Let $\mathbb{A}$ be an effective commutative ring with unity. We denote by $\mathsf{M}_{\mathbb{A}}(d)$ a cost function for multiplying two polynomials $f, g \in \mathbb{A}[x]_{<d}$ by a straight-line program over $\mathbb{A}$. We make the following assumptions:

- $\mathsf{M}_{\mathbb{A}}(d)/d$ is a nondecreasing function in d—this assumption is customary;
- $\mathsf{M}_{\mathbb{A}}$ is *sufficiently close to linear*, in the sense that

$$\frac{\mathsf{M}_{\mathbb{A}}(m\,d)}{m\,d} = O\left(\frac{\mathsf{M}_{\mathbb{A}}(d)}{d}\right) \tag{2.1}$$

  holds whenever $m \leqslant d$. This assumption is less common, but it is only used within Propositions 2.6 and 7.7. Notice that (2.1) is equivalent to $\mathsf{M}_{\mathbb{K}}(m\,d) = O(m\,\mathsf{M}_{\mathbb{K}}(d))$.

For general $\mathbb{A}$, it has been shown in [6] that one may take

$$\mathsf{M}_{\mathbb{A}}(d) = O(d \log d \log\log d) = \tilde{O}(d).$$

If $\mathbb{A}$ has positive characteristic, then it was shown in [23, 24] that one may even take

$$\mathsf{M}_{\mathbb{A}}(d) \;=\; O(d \log d\, 4^{\log^* d}),$$

where $\log^* d := \min\left\{k \in \mathbb{N} : \log \circ \overset{k\times}{\ldots} \circ \log d \leqslant 1\right\}$.

Division Let $f \in \mathbb{A}[x]$ be of degree d_1 and let $g \in \mathbb{A}[x]$ be monic of degree $d_2 \leqslant d_1$. The quotient of the Euclidean division of f by g can be computed by a straight-line program in time $O(\mathsf{M}_{\mathbb{A}}(d_1 - d_2 + 1))$. Given this quotient q, the remainder $r := f - q\,g \in \mathbb{A}[x]_{<d_2}$ can be computed using $O(\mathsf{M}_{\mathbb{A}}(d_2))$ additional operations. If necessary, the degree of r can be determined using at most d_2 further zero-tests.

Gcd It is well known that the gcd of two polynomials F and G in $\mathbb{K}[x]_{\leqslant d}$ can be computed in time $O(\mathsf{M}_{\mathbb{K}}(d) \log d)$; see [21, chapter 11]. But we will need to be more precise in section 5.3 in order to analyze the complexity of "directed inversions in towers". Following the notation of [37], let $R_0 = F$ and $R_1 = G$ be in $\mathbb{K}[x]$ of respective degrees $n_0 := d \geqslant 1$ and $n_1 \leqslant n_0 - 1$, and consider the extended Euclidean sequence defined as follows:

- $C_0 := 1$, $D_0 := 0$, $C_1 := 0$, $D_1 := 1$;
- $R_{i+1} := R_{i-1} - E_i R_i$, $C_{i+1} := C_{i-1} - E_i C_i$, $D_{i+1} := D_{i-1} - E_i D_i$, where $E_i := R_{i-1}$ quo R_i is the quotient of R_{i-1} by R_i.

Consequently we have $R_{i+1} = R_{i-1} \operatorname{rem} R_i$, that is the remainder in the division of R_{i-1} by R_i, and $R_i = C_i F + D_i G$ for all $i \geqslant 0$. The sequence ends after $w-1$ division steps with $R_i \neq 0$ for $i = 0, \ldots, w$, and $R_{w+1} = 0$. The last non-zero polynomial R_w is $\gcd(F, G)$ and the Bézout relation is $R_w = C_w F + D_w G$.

The fast extended Euclidean algorithm applied to R_0 and R_1 does not compute the complete Euclidean sequence, but only the sequence of the quotients $E_1, \ldots, E_w$ in a divide and conquer fashion. This is enough to permit the efficient recovery of the Bézout relation $R_w = C_w F + D_w G$. For $i = 0, \ldots, w+1$, let $n_i := \deg R_i$. We claim that the number of zero-tests in the fast extended Euclidean algorithm is exactly n_0. This claim is a consequence of the two following observations about the fast extended Euclidean algorithm (see for instance [37, Algorithm 18]):

- Computing the quotients E_i does not involve any zero-tests or inversions. On the other hand the determination of n_{i+1} requires testing the coefficients of R_{i+1} from degree $n_i - 1$ to n_{i+1}, which involves $n_i - n_{i+1}$ zero-tests. This totalizes n_0 zero-tests.
- The degrees of the cofactors C_i and D_i, but also of all the entries of the "transition matrices" are explicitly determined from the n_i; see [37, Lemma 10]. So the computation of the transition matrices also does not involve any zero-tests or inversions.

In order to perform polynomial divisions, the inverses of the leading coefficients of the R_i are needed. This requires at most $n_0 = \deg F$ inversions.

To summarize the discussion, the extended gcd of F and G of degree $\leqslant d$ takes $O(\mathsf{M}_{\mathbb{K}}(d) \log d)$ additions, subtractions and products in $\mathbb{K}$, plus $\leqslant d$ zero-tests and $\leqslant d$ inversions. In addition, every inversion of an element is preceded by an unsuccessful zero-test for that element.

2.3. Towers

Elements of a tower $(\mathbb{A}_i)_{i \leqslant t}$ are *in fine* represented by vectors of coordinates in $\mathbb{K}^d$. So additions, subtractions, and products by scalars in $\mathbb{K}$ can be done by straight-line programs with linear costs. We recall the following result, where $\mathsf{m}_{\mathbb{A}_t}$ represents the cost of one product in $\mathbb{A}_t$.

PROPOSITION 2.6. *Under the assumptions of section 2.2, there exists a constant $1 < C \leqslant 3$ such that one product in $\mathbb{A}_t$ can be computed by a straight-line program in time*

$$\mathsf{m}_{\mathbb{A}_t} = O(C^t \mathsf{M}_{\mathbb{K}}(d)).$$

In addition, the product of two polynomials in $\mathbb{A}_t[x]_{<n}$ can be computed by a straight-line program in time

$$\mathsf{M}_{\mathbb{A}_t}(n) = O(C^t \mathsf{M}_{\mathbb{K}}(dn)).$$

Proof. This result is essentially due to Lebreton [35]; see also [29, Proposition 2.4]. □

3. EVALUATION MODELS

In the introduction, we have informally discussed computations with parameters and various strategies for automating case distinctions. In this section, we formalize these ideas in the specific situation of algebraic parameters.

3.1. Parametric frameworks

Let $\mathbb{K}$ be an effective field and let I be an ideal of $\mathbb{K}[x_1,\ldots,x_t]$ such that:

- $\mathbb{A} := \mathbb{K}[x_1,\ldots,x_t]/I$ is a $\mathbb{K}$-algebra of finite dimension $d := \dim_{\mathbb{K}} \mathbb{A}$,
- I is absolutely radical, which means radical over the algebraic closure $\bar{\mathbb{K}}$ of $\mathbb{K}$.

In the context of the present paper, it will be convenient to call $\mathbb{A}$ a *parametric algebra* over $\mathbb{K}$ with t parameters, and I is said to be its *defining ideal*, written $I_{\mathbb{A}}$. The respective images of $x_1,\ldots,x_t$ in $\mathbb{A}$ are denoted by $\alpha_1,\ldots,\alpha_t$; they are regarded as *parameters* over $\mathbb{K}$, subject to the relations in I. Geometrically speaking, the tuple $(\alpha_1,\ldots,\alpha_t)$ represents a generic point in the set $Z_{\mathbb{A}}$ of the d zeros of I in $\bar{\mathbb{K}}^t$. This setting is more general than the one of the introduction, but it is intended to be applied later to ideals I generated by triangular sets.

Let $\mathbb{B}$ be a second parametric algebra over $\mathbb{K}$ with t parameters. As a shorthand, we write $\mathbb{B} \leqslant \mathbb{A}$ whenever $\mathbb{B}$ is a quotient algebra of $\mathbb{A}$. This is the case if, and only if, $I_{\mathbb{B}} \supseteq I_{\mathbb{A}}$, or, equivalently, if $Z_{\mathbb{B}} \subseteq Z_{\mathbb{A}}$. If $\mathbb{B} \leqslant \mathbb{A}$, then we write $\pi_{\mathbb{A}\to\mathbb{B}}$ for the canonical projection $\mathbb{A} \to \mathbb{B}$, and notice that $(\pi_{\mathbb{A}\to\mathbb{B}}(\alpha_1),\ldots,\pi_{\mathbb{A}\to\mathbb{B}}(\alpha_t))$ represents a generic point in the set $Z_{\mathbb{B}}$ of the zeros of $I_{\mathbb{B}}$ in $\bar{\mathbb{K}}^t$. With a slight abuse of notation, $\pi_{\mathbb{A}\to\mathbb{B}}$ is extended in a coefficient-wise manner to $\mathbb{A}[x] \to \mathbb{B}[x]$.

Two parametric algebras $\mathbb{B}_1 \leqslant \mathbb{K}[x_1,\ldots,x_t]$ and $\mathbb{B}_2 \leqslant \mathbb{K}[x_1,\ldots,x_t]$ are said to be *disjoint* if $Z_{\mathbb{B}_1} \cap Z_{\mathbb{B}_2} = \emptyset$. In that case, we have

$$\mathbb{K}[x_1,\ldots,x_t]/(I_{\mathbb{B}_1} \cap I_{\mathbb{B}_2}) \cong \mathbb{B}_1 \oplus \mathbb{B}_2.$$

More generally, given pairwise disjoint $\mathbb{B}_1,\ldots,\mathbb{B}_s \leqslant \mathbb{K}[x_1,\ldots,x_t]$, we have

$$\mathbb{A} := \mathbb{K}[x_1,\ldots,x_t]/(I_{\mathbb{B}_1} \cap \cdots \cap I_{\mathbb{B}_s}) \cong \mathbb{B}_1 \oplus \cdots \oplus \mathbb{B}_s.$$

We will call such a decomposition a *splitting* of $\mathbb{A}$.

Now consider the prime decomposition

$$I = \mathfrak{P}_1 \cap \cdots \cap \mathfrak{P}_s$$

of the defining ideal I of $\mathbb{A}$, and define $\mathbb{E}_i := \mathbb{K}[x_1,\ldots,x_t]/\mathfrak{P}_i$ for $i = 1,\ldots,s$. Then we have $\mathbb{E}_i \leqslant \mathbb{A}$ and there exists a natural isomorphism

$$\mathbb{A} \cong \mathbb{E}_1 \oplus \cdots \oplus \mathbb{E}_s. \tag{3.1}$$

It is convenient to call this relation the *total splitting* of $\mathbb{A}$. The corresponding zero-set $Z_{\mathbb{A}}$ is a disjoint union:

$$Z_{\mathbb{A}} = Z_{\mathbb{E}_1} \amalg \ldots \amalg Z_{\mathbb{E}_s}.$$

Given a zero-divisor $b \in \mathbb{A}$, we have

$$I = I_1 \cap I_2, \text{ where } I_1 := I + (b) \text{ and } I_2 := I : (b).$$

Indeed, we clearly have $I \subseteq I_1 \cap I_2$. Conversely, given $f = u + bv \in I : (b)$ with $u \in I$, we have $b^2 v \in I$, whence $bv \in I$ since I is radical, and $f \in I$. Setting

$$\begin{aligned} \mathbb{B}_1 &:= \mathbb{K}[x_1,\ldots,x_t]/I_1 \cong \mathbb{A}/(b) \\ \mathbb{B}_2 &:= \mathbb{K}[x_1,\ldots,x_t]/I_2 \cong \mathbb{A}[b^{-1}], \end{aligned}$$

we obtain a natural isomorphism

$$\mathbb{A} \cong \mathbb{B}_1 \oplus \mathbb{B}_2.$$

Such a decomposition is called an *atomic splitting* of $\mathbb{A}$. We have

$$Z_{\mathbb{A}} = Z_{\mathbb{B}_1} \amalg Z_{\mathbb{B}_2}.$$

From an effective point of view, the computation of total splittings requires an algorithm for polynomial factorization, whereas atomic splittings can be computed using standard techniques from effective polynomial algebra, such as Gröbner bases, triangular sets, or geometric resolutions.

Let us now describe an abstract way to formalize computations in parametric algebras modulo splittings. Consider a set $\mathcal{F}$ of parametric algebras over $\mathbb{K}$. We say that $\mathcal{F}$ is a *parametric framework* if the following conditions are satisfied:

- Each $\mathbb{A} \in \mathcal{F}$ is an effective $\mathbb{K}$-algebra.
- Given $a \in \mathbb{A} \in \mathcal{F}$, we can test whether a is invertible and compute a^{-1} if so.
- Given $\mathbb{A}, \mathbb{B} \in \mathcal{F}$ with $\mathbb{B} \leqslant \mathbb{A}$, we have an algorithm for the projection $\pi_{\mathbb{A} \to \mathbb{B}}$.
- Given $b \in \mathbb{A} \in \mathcal{F}$, we can effectively compute $\mathbb{A}/(b) \in \mathcal{F}$ and $\mathbb{A}[b^{-1}] \in \mathcal{F}$.
- For any $\mathbb{A} \cong \mathbb{B}_1 \oplus \cdots \oplus \mathbb{B}_s$ with $\mathbb{A}, \mathbb{B}_1, \ldots, \mathbb{B}_s \in \mathcal{F}$, we have algorithms for both directions of this isomorphism.

Using Gröbner basis techniques, the above conditions are in particular verified if $\mathcal{F}$ is the set of all parametric algebras $\mathbb{A}$ that are explicitly given by a finite set of generators of $I_{\mathbb{A}}$. Indeed, a Gröbner basis of $I_{\mathbb{A}}$ induces a basis of $\mathbb{A}$ over $\mathbb{K}$ along with the multiplication matrices by $\alpha_1, \ldots, \alpha_t$, so the above operations boil down to linear algebra. This observation is convenient from a conceptual point of view, but, of course, not very efficient from the asymptotic complexity perspective. Sections 4, 5, and 7 are devoted to more specific parametric frameworks that allow for asymptotically faster implementations.

3.2. Unpermissive evaluation

From now on, $\mathbb{A}$ denotes a parametric algebra, and we will use the notations from section 2.1 for computation trees. The *unpermissive evaluation* of a computation tree T with input nodes $\mathcal{I}$ as above at $a = (a_1, \ldots, a_{|\mathcal{I}|}) \in \mathbb{A}^{|\mathcal{I}|}$ is defined as follows. Informally speaking, the tree is evaluated in the usual way over $\mathbb{A}$ concerning ring operations, but the evaluation aborts whenever a zero-divisor needs to be inverted or tested to zero.

More precisely, we write $\bar{P}(T; a)$ for the *unpermissive path*, and $\bar{\mathcal{E}}$ for the *unpermissive evaluation function*. We introduce a new output value, written $\mathsf{inconsistent}$, in order to indicate that the evaluation process detects that the current algebra $\mathbb{A}$ is not a field. We define $\bar{\mathcal{E}}(v; a)$ by induction:

- If the current node v of the path is a computation node of the form $(\mathsf{invert}; u)$, then we distinguish the following cases:
  - If $\bar{\mathcal{E}}(u; a)$ is zero in $\mathbb{A}$, then the path ends at v with $\bar{\mathcal{E}}(T; a) := \bar{\mathcal{E}}(v; a) := \mathsf{undefined}$.
  - If $\bar{\mathcal{E}}(u; a)$ is invertible, then $\bar{\mathcal{E}}(v; a) := \bar{\mathcal{E}}(u; a)^{-1}$ and the next node of the path becomes the successor of v.
  - Otherwise, $\bar{\mathcal{E}}(u; a)$ is a zero-divisor in $\mathbb{A}$, and the evaluation aborts; the path ends at v, we set $\bar{\mathcal{E}}(T; a) := \bar{\mathcal{E}}(v; a) := \mathsf{inconsistent}$, and record the zero-divisor $\bar{\mathcal{E}}(u; a)$ for future use.
- If the current node v in the path is a branching node $(\mathsf{zero?}; u)$, then we distinguish the following cases:
  - If $\bar{\mathcal{E}}(u; a)$ is zero in $\mathbb{A}$, then $\bar{\mathcal{E}}(v; a) := \mathsf{true}$, and the next node of the path becomes the right successor of v.

- If $\bar{\mathcal{E}}(u;a)$ is invertible in $\mathbb{A}$, then we set $\bar{\mathcal{E}}(v;a) := \mathsf{false}$, and the next node of the path becomes the left successor of v.
- Otherwise, $\bar{\mathcal{E}}(u;a)$ is a zero-divisor in $\mathbb{A}$, and the evaluation aborts; the path ends at v, we set $\bar{\mathcal{E}}(T;a) := \bar{\mathcal{E}}(v;a) := \mathsf{inconsistent}$, and record the zero-divisor $\bar{\mathcal{E}}(u;a)$ for future use.

- For the other types of nodes, the evaluation rules remain the same as for the standard evaluation described in section 2.1.

LEMMA 3.1. *With the above notations, let $\mathbb{B} \leqslant \mathbb{A}$. If $\bar{\mathcal{E}}(T;a) \neq \mathsf{inconsistent}$, then the evaluation paths $\bar{P}(T;a)$ and $P(T;\pi_{\mathbb{A}\to\mathbb{B}}(a))$ coincide and $\pi_{\mathbb{A}\to\mathbb{B}}(\bar{\mathcal{E}}(T;a)) = \mathcal{E}(T;\pi_{\mathbb{A}\to\mathbb{B}}(a))$ holds, under the natural convention that $\pi_{\mathbb{A}\to\mathbb{B}}(\mathsf{undefined}) := \mathsf{undefined}$.*

Proof. The proof is straightforward from the definitions. □

3.3. Panoramic values

DEFINITION 3.2. *Given an input $a = (a_1, \ldots, a_{|\mathcal{J}|}) \in \mathbb{A}^{|\mathcal{J}|}$ for a computation tree T, we say that*

$$\mathbb{A} \cong \mathbb{D}_1 \oplus \cdots \oplus \mathbb{D}_\ell$$

*is a **panoramic splitting** of $\mathbb{A}$ for the pair (T,a) if $\bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{D}_i}(a)) \neq \mathsf{inconsistent}$ for $i = 1,\ldots,\ell$. A **panoramic value** of T at a is a set of pairs $\{(\mathbb{D}_1,b_1),\ldots,(\mathbb{D}_\ell,b_\ell)\}$ such that $\mathbb{D}_1 \oplus \cdots \oplus \mathbb{D}_\ell$ is a panoramic splitting of $\mathbb{A}$ and $b_i := \bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{D}_i}(a))$ for $i = 1,\ldots,\ell$.*

LEMMA 3.3. *Let $\mathbb{A} \cong \mathbb{E}_1 \oplus \cdots \oplus \mathbb{E}_s$ denote the total splitting of $\mathbb{A}$, and let $\{(\mathbb{D}_1,b_1),\ldots,(\mathbb{D}_\ell,b_\ell)\}$ be a panoramic value of T at a. Then, for each $i \in \{1,\ldots,s\}$, we have*

$$\bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{E}_i}(a)) = \mathcal{E}(T;\pi_{\mathbb{A}\to\mathbb{E}_i}(a)) = \pi_{\mathbb{D}_j\to\mathbb{E}_i}(b_j),$$

where $j \in \{1,\ldots,\ell\}$ is the unique integer such that $\mathbb{E}_i \leqslant \mathbb{D}_j$.

Proof. Since $\mathbb{E}_i$ is a field, any non-zero element in $\mathbb{E}_i$ is invertible. Consequently,

$$\bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{E}_i}(a)) \neq \mathsf{inconsistent},$$

and Lemma 3.1 implies that $\bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{E}_i}(a)) = \mathcal{E}(T;\pi_{\mathbb{A}\to\mathbb{E}_i}(a))$. On the other hand,

$$b_j = \bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{D}_i}(a)) \neq \mathsf{inconsistent},$$

and Lemma 3.1 implies that

$$\pi_{\mathbb{D}_j\to\mathbb{E}_i}(b_j) = \mathcal{E}(T;\pi_{\mathbb{A}\to\mathbb{E}_i}(a)).$$ □

Example 3.4. Consider the computation tree T from Example 2.1 for some $\kappa \in \mathbb{N}$, and let

$$\mu(x) := (x^2 - p_1)\cdots(x^2 - p_\delta),$$

$I := (\mu(x))$, and $\mathbb{A} := \mathbb{Q}[x]/I$, with $\delta > \kappa$. If α represents the image of x in $\mathbb{A}$, then a possible panoramic splitting of T at α is

$$\mathbb{A} \cong \mathbb{D}_1 \oplus \cdots \oplus \mathbb{D}_\ell,$$

where

- $\ell := \kappa + 1$,

- $\mathbb{D}_i := \mathbb{Q}[x]/(x^2-p_i)$ and $\bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{D}_i}(\alpha)) = i$, for $i=1,\ldots,\kappa$,
- $\mathbb{D}_\ell := \mathbb{Q}[x]/((x^2-p_{\kappa+1})\cdots(x^2-p_\delta))$ and $\bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{D}_\ell}(\alpha)) = 0$.

Notice that the associated primes of I are $\mathfrak{P}_i = (x^2-p_i)$ for $i=1,\ldots,\delta$, and the total splitting of $\mathbb{A}$ is

$$\mathbb{A} \cong \bigoplus_{i=1}^{\delta} \mathbb{Q}[x]/(x^2-p_i).$$

The following naive algorithm for the computation of panoramic values was already sketched in the introduction:

Algorithm 3.1

Input. A computation tree T and an evaluation point $a \in \mathbb{A}^{|\mathcal{I}|}$.

Output. A panoramic value of T at a.

1. Compute the unpermissive evaluation $\bar{\mathcal{E}}(T;a)$.
2. If $\bar{\mathcal{E}}(T;a) \neq$ inconsistent, then return $\{(\mathbb{A}, \bar{\mathcal{E}}(T;a))\}$.
3. Otherwise a non-trivial zero-divisor $b \in \mathbb{A}$ has been recorded, which allows us to compute proper subalgebras $\mathbb{B}_1,\ldots,\mathbb{B}_l$ and $k \in \{2,\ldots,l-1\}$ such that $\mathbb{A}/(b) \cong \mathbb{B}_1 \oplus \cdots \oplus \mathbb{B}_k$ and $\mathbb{A}[b^{-1}] \cong \mathbb{B}_{k+1} \oplus \cdots \oplus \mathbb{B}_l$.
4. Recursively apply the algorithm to $\pi_{\mathbb{A}\to\mathbb{B}_i}(a)$ for $i=1,\ldots,l$ and return the union the panoramic values obtained in this way.

PROPOSITION 3.5. *Algorithm 3.1 is correct.*

Proof. The algorithm finishes since $\mathbb{A}$ is finite dimensional. As for the correctness it suffices to verify that the output in step 2 is correct, which is straightforward from the definitions. □

3.4. Directed operations

From now we focus on the development of a faster alternative for Algorithm 3.1. The idea is to impose a finer control over the dimension of the components of splittings. Informally speaking, computation trees over $\mathbb{A}$ will be evaluated entirely, while restricting operations to suitable subalgebras of $\mathbb{A}$: when a zero test or an inversion occurs for a value a, the current subalgebra is further restricted to its largest subalgebra where the projection of a is either invertible or zero.

In order to formalize this idea, we need some definitions and the underlying directed arithmetic operations.

DEFINITION 3.6. *Given a computation tree T, we say that a splitting*

$$\mathbb{A} \cong \mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$$

*is **directed** if*

- $\dim_{\mathbb{K}} \mathbb{D} \geqslant 1$ *and* $\dim_{\mathbb{K}} \mathbb{H}_i \geqslant 1$ *for* $i=1,\ldots,\ell$,
- $2\dim_{\mathbb{K}} \mathbb{H}_i \leqslant \dim_{\mathbb{K}} \mathbb{A}$ *for* $i=1,\ldots,\ell$.

*A **directed evaluation** of T takes as input:*

- *a directed splitting of* $\mathbb{A} \cong \mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$,

- *a point* $a \in \mathbb{D}^{|\mathcal{J}|}$,

and returns

- *a refined directed splitting of* $\mathbb{A} \cong \tilde{\mathbb{D}} \oplus \tilde{\mathbb{H}}_1 \oplus \cdots \oplus \tilde{\mathbb{H}}_s \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$,
- $\bar{\mathcal{E}}(T; \pi_{\mathbb{D} \to \tilde{\mathbb{D}}}(a))$ *such that* $\bar{\mathcal{E}}(T; \pi_{\mathbb{D} \to \tilde{\mathbb{D}}}(a)) \neq \mathsf{inconsistent}$.

We introduce a *directed variant of the inversion* in $\mathbb{A}$, that takes a directed splitting $\mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ as input, the value a in $\mathbb{D}$ to be inverted, and that returns a directed splitting $\tilde{\mathbb{D}} \oplus \tilde{\mathbb{H}}_1 \oplus \cdots \oplus \tilde{\mathbb{H}}_s \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ such that

- $\pi_{\mathbb{D} \to \tilde{\mathbb{D}}}(a)$ is either zero or invertible;
- The result of the directed inversion of a is respectively $\mathsf{undefined}$ or $\pi_{\mathbb{D} \to \tilde{\mathbb{D}}}(a)^{-1}$.

We also introduce a *directed variant of the zero-test* in $\mathbb{A}$, that takes a directed splitting $\mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ and the value a in $\mathbb{D}$ to be tested as input, and that returns a directed splitting $\tilde{\mathbb{D}} \oplus \tilde{\mathbb{H}}_1 \oplus \cdots \oplus \tilde{\mathbb{H}}_s \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ such that

- $\pi_{\mathbb{D} \to \tilde{\mathbb{D}}}(a)$ is either zero or invertible;
- If $\pi_{\mathbb{D} \to \tilde{\mathbb{D}}}(a)$ is invertible, then $\pi_{\mathbb{D} \to \tilde{\mathbb{D}}}(a)^{-1}$ has been computed and recorded;
- The result is $\mathsf{zero?}(\pi_{\mathbb{D} \to \tilde{\mathbb{D}}}(a))$.

Remark 3.7. The above specifications leave room for some flexibility for the precise implementation of zero-tests and inversions: in general, the conditions $2 \dim_{\mathbb{K}} \mathbb{H}_i \leqslant \dim_{\mathbb{K}} \mathbb{A}$ do not imply uniqueness of the output splittings.

Remark 3.8. We could have introduced directed variants of additions, subtractions, and products in parametric algebras, exactly in the same way as for the zero-test and the inversion. This could actually be useful in practice. However, in order to keep our exposition simple, we prefer to use straight-line programs over $\mathbb{K}$ for these operations, so splittings never occur.

3.5. Directed evaluation

Let us now define the directed evaluation of a tree T. As input, we take a directed splitting $\mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ and $a \in \mathbb{D}^{|\mathcal{J}|}$. Informally speaking, this evaluation simply applies the above directed operations in sequence, so we evaluate the entire computation tree while restricting the current algebra when encountering zero-tests and inversions. The list of the residual subalgebras is maintained throughout the evaluation process. For consistency, the evaluation takes a directed splitting $\mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ as input, but only the summand $\mathbb{D}$ will be decomposed during the evaluation process.

Formally speaking, the directed evaluation path is written $\vec{P}(T; a)$. To each node $v \in \vec{P}(T; a)$, we both associate a directed splitting

$$\vec{\mathcal{D}}(v; a) = (\mathbb{D}_v, \mathbb{H}_{v,1}, \ldots, \mathbb{H}_{v,\ell_v})$$

of $\mathbb{A}$, so

$$\mathbb{A} \cong \mathbb{D}_v \oplus \mathbb{H}_{v,1} \oplus \cdots \oplus \mathbb{H}_{v,\ell_v},$$

and a value

$$\vec{\mathcal{E}}(v; a) \in \mathbb{D}_v \cup \{\mathsf{false}, \mathsf{true}, \mathsf{undefined}\} \cup \bigcup_{v \in O} \mathbb{D}_v^{m(v)}.$$

The directed evaluation of T is defined inductively as follows:

- The path $\vec{P}(T;a)$ starts with the input nodes $u_1,\ldots,u_{|\mathcal{I}|}$ of $\mathcal{I}$. We set $\vec{\mathcal{D}}(u_i;a) := (\mathbb{D},\mathbb{H}_1,\ldots,\mathbb{H}_\ell)$ and $\vec{\mathcal{E}}(u_i;a) := a_i$, for $i=1,\ldots,|\mathcal{I}|$. The next node of the path is set to the successor of $u_{|\mathcal{I}|}$.
- If the current node v of the path is the root of the path, and is a computation node $(\mathsf{op};u_1,\ldots,u_m)$, then we necessarily have $\mathcal{I}=\emptyset$ and $m=0$, we set $\vec{\mathcal{D}}(v;a) := (\mathbb{D},\mathbb{H}_1,\ldots,\mathbb{H}_\ell)$ and $\vec{\mathcal{E}}(v;a) := \mathsf{op}()$. The next node of the path is set to the successor of v.
- If the current node v of the path is a computation node $(\mathsf{op};u_1,\ldots,u_m)$ with $\mathsf{op}\in \mathbb{K}_0\cup\mathbb{K}_1\cup\{+,-,\times\}$, whose direct predecessor is w, then we set $\vec{\mathcal{D}}(v;a) := \vec{\mathcal{D}}(w;a)$ and

$$\vec{\mathcal{E}}(v;a) := \mathsf{op}(\pi_{\mathbb{D}_{u_1}\to\mathbb{D}_w}(\vec{\mathcal{E}}(u_1;a)),\ldots,\pi_{\mathbb{D}_{u_m}\to\mathbb{D}_w}(\vec{\mathcal{E}}(u_m;a))).$$

  The next node of the path is set to the successor of v.
- If the current node v of the path is a computation node $(\mathsf{invert};u)$, whose direct predecessor is w, then we apply the directed inversion to $e := \pi_{\mathbb{D}_u\to\mathbb{D}_w}(\vec{\mathcal{E}}(u;a))$ for the splitting $\vec{\mathcal{D}}(w;a)$. This yields a new splitting $\mathbb{A}\cong\mathbb{D}_v\oplus\mathbb{H}_{v,1}\oplus\cdots\oplus\mathbb{H}_{v,\ell_v}$ that we assign to $\vec{\mathcal{D}}(v;a)$.
  - If $\pi_{\mathbb{D}_w\to\mathbb{D}_v}(e)=0$, then we set $\vec{\mathcal{E}}(T;a) := \vec{\mathcal{E}}(v;a) := \mathsf{undefined}$, and the path ends at v.
  - Otherwise $\vec{\mathcal{E}}(v;a) := \pi_{\mathbb{D}_w\to\mathbb{D}_v}(e)^{-1}$, and the next node of the path becomes the successor of v.
- If the current node v of the path is a branching node $(\mathsf{zero?};u)$, whose direct predecessor is w, then we apply the directed zero-test to $e := \pi_{\mathbb{D}_u\to\mathbb{D}_w}(\vec{\mathcal{E}}(u;a))$ for the splitting $\vec{\mathcal{D}}(w;a)$. This yields a new splitting $\mathbb{A}\cong\mathbb{D}_v\oplus\mathbb{H}_{v,1}\oplus\cdots\oplus\mathbb{H}_{v,\ell_v}$, that we assign to $\vec{\mathcal{D}}(v;a)$.
  - If $\pi_{\mathbb{D}_w\to\mathbb{D}_v}(e)=0$, then $\vec{\mathcal{E}}(v;a) := \mathsf{true}$, and the current node of the path becomes the right successor of v.
  - Otherwise, $\vec{\mathcal{E}}(v;a) := \mathsf{false}$, and the current node of the path becomes the left successor of v. We also record the inverse $\pi_{\mathbb{D}_w\to\mathbb{D}_v}(e)^{-1}$.
- If the current node v of the path is an output node $(u_1,\ldots,u_m)$, whose predecessor is w, then we set $\vec{\mathcal{D}}(T;a) := \vec{\mathcal{D}}(v;a) := \vec{\mathcal{D}}(w;a)$ and

$$\vec{\mathcal{E}}(T;a) := \vec{\mathcal{E}}(v;a) := (\pi_{\mathbb{D}_{u_1}\to\mathbb{D}_w}(\vec{\mathcal{E}}(u_1;a)),\ldots,\pi_{\mathbb{D}_{u_m}\to\mathbb{D}_w}(\vec{\mathcal{E}}(u_m,a))).$$

Whenever $u,v\in\vec{P}(T;a)$ are such that v is a successor of u, the splittings of $\mathbb{A}$ at u and v satisfy $\mathbb{D}_v\leqslant\mathbb{D}_u$. It can be checked by induction over the path $\vec{P}(T;a)$ that $\vec{\mathcal{D}}(T;a)$ and $\vec{\mathcal{E}}(T;a)$ are well defined.

PROPOSITION 3.9. *Let $\mathbb{A}\cong\tilde{\mathbb{D}}\oplus\tilde{\mathbb{H}}_1\oplus\cdots\oplus\tilde{\mathbb{H}}_s\oplus\mathbb{H}_1\oplus\cdots\oplus\mathbb{H}_\ell$ represent the directed splitting $\vec{\mathcal{D}}(T;a)$ returned by the directed evaluation of T at a as above. Then, for all $\mathbb{E}\leqslant\tilde{\mathbb{D}}$, we have*

$$\pi_{\tilde{\mathbb{D}}\to\mathbb{E}}(\vec{\mathcal{E}}(T;a)) = \bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{E}}(a)).$$

Proof. We verify by induction on paths that $\vec{P}(T;a)$ and $\bar{P}(T;\pi_{\mathbb{A}\to\mathbb{E}}(a))$ coincide, and that $\pi_{\mathbb{D}_v\to\mathbb{E}}(\vec{\mathcal{E}}(v;a)) = \bar{\mathcal{E}}(v;\pi_{\mathbb{A}\to\mathbb{E}}(a))$ holds for all $v\in\vec{P}(T;a)$ that is not a branching node. □

Example 3.10. Let us again consider the computation tree T from Example 2.1, and let

$$\mu(x) := (x^2-p_1)\cdots(x^2-p_\delta),$$

$I := (\mu(x))$, and $\mathbb{A} := \mathbb{Q}[x]/I$, with $\delta > \kappa$. Consider the directed evaluation of T at $a = \alpha$, where α represents the image of x in $\mathbb{A}$. The first zero-test that we encounter is $\mathsf{zero?}(a^2-p_1)$, which yields the directed splitting

$$\mathbb{A} \cong \mathbb{Q}[x]/((x^2-p_2)\cdots(x^2-p_\delta)) \oplus \mathbb{Q}[x]/(x^2-p_1).$$

After the κ zero-tests $\mathsf{zero?}(a^2-p_1), \ldots, \mathsf{zero?}(a^2-p_\kappa)$, we end up with the directed decomposition

$$\mathbb{A} \cong \mathbb{Q}[x]/((x^2-p_{\kappa+1})\cdots(x^2-p_\delta)) \oplus \mathbb{Q}[x]/(x^2-p_\kappa) \oplus \cdots \oplus \mathbb{Q}[x]/(x^2-p_1)$$

and the value $\vec{\mathcal{E}}(T;\alpha) = 0$. This is due to the requirement that directed evaluation always privileges the "branch of highest degree". If $\delta = \kappa + 1$, then the last zero-test potentially leads to another directed decomposition

$$\mathbb{A} \cong \mathbb{Q}[x]/(x^2-p_\kappa) \oplus \mathbb{Q}[x]/(x^2-p_{\kappa+1}) \oplus \mathbb{Q}[x]/(x^2-p_{\kappa-1}) \oplus \cdots \oplus \mathbb{Q}[x]/(x^2-p_1)$$

and the value $\vec{\mathcal{E}}(T;\alpha) = \kappa$. In general, we recall that directed evaluation is a non-deterministic process: the output may depend on internal splitting choices that are made during the directed zero-tests and inversions.

3.6. Fast panoramic evaluation

We are now in a position to state a central algorithm of this paper, which performs panoramic evaluations using directed ones. The presentation is abstract, in terms of general parametric frameworks. In the remainder of the paper we will study various concrete instantiations of this algorithm.

Algorithm 3.2

Input. A computation tree T and an evaluation point $a \in \mathbb{A}^{|\mathcal{J}|}$.

Output. A panoramic value of T at a.

1. Perform the directed evaluation of T at a for the trivial directed splitting $(\mathbb{A})$ of $\mathbb{A}$. Let $\mathbb{A} \cong \mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ be the directed splitting obtained in return.
2. Compute the projections $\pi_{\mathbb{A}\to\mathbb{H}_i}(a)$ for $i = 1, \ldots, \ell$.
3. Recursively call the algorithm in order to evaluate T at $\pi_{\mathbb{A}\to\mathbb{H}_i}(a)$ for $i = 1, \ldots, \ell$.
4. Return the union of $\{(\mathbb{D}, \vec{\mathcal{E}}(T;a))\}$ and of the panoramic values obtained in step 3.

PROPOSITION 3.11. *Algorithm 3.2 is correct.*

Proof. Algorithm 3.2 finishes because the dimension of the input algebra strictly decreases throughout the recursive calls in step 3. At the end of step 1, we have

$$\vec{\mathcal{E}}(T;a) = \bar{\mathcal{E}}(T;\pi_{\mathbb{A}\to\mathbb{D}}(a)) \neq \mathsf{inconsistent},$$

by Proposition 3.9. Therefore the singleton $\{(\mathbb{D}, \vec{\mathcal{E}}(T;a))\}$ is a panoramic value of T at $\pi_{\mathbb{A}\to\mathbb{D}}(a)$. We conclude by observing that the union of the panoramic values in step 4 gives a panoramic value of T at a. □

Example 3.12. Example 3.10 illustrates step 1 of Algorithm 3.2, for which we obtain $\mathbb{D} = \mathbb{Q}[x]/((x^2-p_{\kappa+1})\cdots(x^2-p_\delta))$, $\vec{\mathcal{E}}(T;\alpha) = 0$ and $\mathbb{H}_i = \mathbb{Q}[x]/(x^2-p_i)$ for $i = 1,\ldots,\kappa$. After the κ recursive calls in step 3, we obtain the panoramic value

$$\{(\mathbb{Q}[x]/((x^2-p_\kappa)\cdots(x^2-p_\delta)),0),(\mathbb{Q}[x]/(x^2-p_1),1),\ldots,(\mathbb{Q}[x]/(x^2-p_\kappa),\kappa)\}.$$

In the next sections, Algorithm 3.2 will lead to nearly linear asymptotic complexity bounds. So it turns out to be much faster than the naive Algorithm 3.1.

3.7. Delayed reductions

From the complexity point of view, there is still a problem with naive implementations of Algorithm 3.2: the directed evaluation approach involves many potentially expensive conversions of the form $\pi_{\mathbb{D}_u\to\mathbb{D}_w}(\vec{\mathcal{E}}(u;a))$. A general solution to this issue is tedious, but for the concrete parametric frameworks from this paper, a natural and efficient approach is to delay these conversions. This can be done at the price of performing arithmetic operations in the input algebra instead of its subalgebras.

Consider for instance the simplest case when $\mathbb{A} = \mathbb{K}[x]/(\mu(x))$, where μ is monic of degree d. Let $\mathbb{D} = \mathbb{K}[x]/(\nu(x)) \leqslant \mathbb{A}$ with $\nu \mid \mu$ and $\deg \nu \geqslant d/2$. Then one projection $\pi_{\mathbb{A}\to\mathbb{D}}$ corresponds to a division by ν of cost $O(\mathsf{M}_{\mathbb{K}}(d))$, when using the standard representation for elements in $\mathbb{D}$. This means that additions during the directed evaluation of a tree typically admit a cost $O(\mathsf{M}_{\mathbb{K}}(d))$ instead of $O(d)$, which is suboptimal.

A natural remedy to this problem is to delay reductions modulo ν. This strategy naturally fits in our setting of parametric frameworks through the use of redundant representations. More precisely, consider some $\mathbb{A} \in \mathcal{F}$ within a parametric framework $\mathcal{F}$. Then we may define a new parametric framework $\mathcal{F}_{\mathbb{A}}$ whose objects are algebras $\mathbb{B} \leqslant \mathbb{A}$ in $\mathcal{F}$, with this modification that elements in $\mathbb{B}$ are represented by elements in $\mathbb{A}$. Given $\mathbb{B}' \leqslant \mathbb{B}$, this means that projections $\pi_{\mathbb{B}\to\mathbb{B}'}$ are free of charge in $\mathcal{F}_{\mathbb{A}}$. The arithmetic operations $\mathsf{op} \in \mathbb{K}_0 \cup \mathbb{K}_1 \cup \{+,-,\times\}$ on elements in $\mathbb{B}$ are also performed in $\mathbb{A}$. The inversion of $b \in \mathbb{B}$ is done by computing the projection $b' = \pi_{\mathbb{A}\to\mathbb{B}}(b)$ in $\mathcal{F}$ and then performing the inversion of b' in $\mathbb{B}$; zero-tests are done similarly.

Of course, at the end of a directed evaluation that uses this kind of redundant representations, we may wish to convert the result to the standard representation in $\mathcal{F}$. This can be done by inserting a "finalization" step at the end of step 1 in Algorithm 3.2.

4. COMPLEXITY OF UNIVARIATE PANORAMIC EVALUATION

This section is devoted to parametric algebras with one algebraic parameter, that is of the form $\mathbb{A} = \mathbb{K}[x]/(\mu_{\mathbb{A}}(x))$ with $\mu_{\mathbb{A}}$ monic, separable, and of degree d. For our complexity bounds it will be convenient to assume that $d \geqslant 2$.

Elements in $\mathbb{A}$ are represented as remainder classes $a = A \bmod \mu_{\mathbb{A}}$ with $A \in \mathbb{K}[x]_{<d}$. The univariate situation is notably simple but already useful enough to be treated separately. We first specify the parametric framework and the directed operations, then we analyze the cost of our fast panoramic evaluation, and finally we compare the directed evaluation strategy with dynamic evaluation.

For efficiency reasons, we further assume that $\mu_{\mathbb{A}}$ is *explicitly separable* in the sense that we are given the cofactors $\theta_{\mathbb{A}}$ and $\xi_{\mathbb{A}}$ in $\mathbb{K}[x]_{<d}$ in the Bézout relation

$$1 = \theta_{\mathbb{A}}\,\mu'_{\mathbb{A}} + \xi_{\mathbb{A}}\,\mu_{\mathbb{A}}. \tag{4.1}$$

Computing this relation from the outset only requires $O(\mathsf{M}_{\mathbb{K}}(d)\log d)$ operations in $\mathbb{K}$.

4.1. Univariate parametric framework

Let us first specify the required operations of the univariate parametric framework, and explicit their complexity.

- Additions and subtractions in $\mathbb{A}$ can clearly be performed by straight-line programs in linear time $O(d)$, whereas multiplications take time $O(\mathsf{M}_{\mathbb{K}}(d))$.
- An element $a = A \bmod \mu_{\mathbb{A}} \in \mathbb{A}$ with $A \in \mathbb{K}[x]_{<d}$ is invertible if and only if $\gcd(A, \mu_{\mathbb{A}}) = 1$ and, in that case, a^{-1} can be computed using the extended Euclidean algorithm. Both the test and the computation of a^{-1} can be done by a computation tree in time $O(\mathsf{M}_{\mathbb{K}}(d) \log d)$.
- We have $\mathbb{B} \leqslant \mathbb{A}$ if and only if $\mu_{\mathbb{B}} \mid \mu_{\mathbb{A}}$. In that case, we have

$$\pi_{\mathbb{A} \to \mathbb{B}}(A \bmod \mu_{\mathbb{A}}) = (A \bmod \mu_{\mathbb{A}}) \bmod \mu_{\mathbb{B}}$$

  for all $A \in \mathbb{K}[x]_{<d}$; the projection $\pi_{\mathbb{A} \to \mathbb{B}}$ can thus be computed in time $O(\mathsf{M}_{\mathbb{K}}(d))$ by a straight-line program (below, we will rather use the technique of delayed reductions from section 3.7 in order to avoid projections altogether).
- Given $b = B \bmod \mu_{\mathbb{A}} \in \mathbb{A} \setminus \{0\}$ with $B \in \mathbb{K}[x]_{<d}$, we have $\mu_{\mathbb{A}/(b)} = \gcd(\mu_{\mathbb{A}}, B)$ and $\mu_{\mathbb{A}[b^{-1}]} = \mu_{\mathbb{A}} / \gcd(\mu_{\mathbb{A}}, B)$. This shows that we can compute $\mathbb{A}/(b)$ and $\mathbb{A}[b^{-1}]$ in time $O(\mathsf{M}_{\mathbb{K}}(d) \log d)$ using computation trees.
- Given univariate parametric algebras $\mathbb{B}_1, \ldots, \mathbb{B}_s \leqslant \mathbb{A}$, we have $\mathbb{A} \cong \mathbb{B}_1 \oplus \cdots \oplus \mathbb{B}_s$ if, and only if, $\mu_{\mathbb{A}} = \mu_{\mathbb{B}_1} \cdots \mu_{\mathbb{B}_s}$ (which in particular implies that the $\mu_{\mathbb{B}_i}$ are pairwise coprime). In this case, the isomorphism $\mathbb{A} \cong \mathbb{B}_1 \oplus \cdots \oplus \mathbb{B}_s$ in both directions can be computed in time $O(\mathsf{M}_{\mathbb{K}}(d) \log s)$ by using the usual subproduct tree of $\mu_{\mathbb{B}_1}, \ldots, \mu_{\mathbb{B}_s}$; see [21, chapter 10].

The directed zero-tests and inversions of an element $a \in \mathbb{A}$ with preimage A can be computed in time $O(\mathsf{M}_{\mathbb{K}}(d) \log d)$, as follows:

- The input is a directed splitting of $\mathbb{A}$,

$$\mathbb{A} \cong \mathbb{K}[x]/(\mu_{\mathbb{D}}(x)) \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell,$$

  the Bézout relation $1 = \theta_{\mathbb{D}} \mu'_{\mathbb{D}} + \xi_{\mathbb{D}} \mu_{\mathbb{D}}$, and an element $a \in \mathbb{K}[x]/(\mu_{\mathbb{D}}(x))$ with preimage $A \in \mathbb{K}[x]_{<\deg \mu_{\mathbb{D}}}$.
- If $A = 0$, then the result of the zero-test is true and the result of the inversion is undefined. The input directed splitting is left unchanged in return.
- We compute $g := \gcd(A, \mu_{\mathbb{D}})$ along with the Bézout relation $g = UA + V\mu_{\mathbb{D}}$.
- If $g = 1$, then the result of the zero-test is true and the result of the inversion is U. The input directed splitting is left unchanged in return.
- We compute $h := \mu_{\mathbb{D}}/g$, and deduce the Bézout relations of g' and g, and of h' and h by Lemma 4.1 below.
- If $2 \deg h \leqslant \deg \mu_{\mathbb{A}}$ then we return the directed splitting

$$\mathbb{A} \cong \mathbb{K}[x]/(g(x)) \oplus \mathbb{K}[x]/(h(x)) \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$$

  and the image of a is zero in $\mathbb{K}[x]/(g(x))$. The value of the zero-test is true, and the result of the inversion is undefined.
- Otherwise, we return the splitting

$$\mathbb{A} \cong \mathbb{K}[x]/(h(x)) \oplus \mathbb{K}[x]/(g(x)) \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$$

and the image of a in $\mathbb{K}[x]/(h(x))$ is invertible. The latter splitting is actually directed since $2\deg h > \deg \mu_{\mathbb{A}}$ implies

$$2\deg g = 2(\deg \mu_{\mathbb{D}} - \deg h) \leqslant 2(\deg \mu_{\mathbb{A}} - \deg h) < \deg \mu_{\mathbb{A}}.$$

The inverse of a modulo h is obtained as follows:

- From the Bézout relation $1 = \theta_{\mathbb{D}}\,\mu'_{\mathbb{D}} + \xi_{\mathbb{D}}\,\mu_{\mathbb{D}}$, we deduce $1 = \theta_{\mathbb{D}}\,(g'h + gh') + \xi_{\mathbb{D}}\,\mu_{\mathbb{D}}$ and then
$$1 = \theta_{\mathbb{D}}\, g h' \bmod h.$$
- From the Bézout relation $g = UA + V\,\mu_{\mathbb{D}}$, we get $g = UA \bmod h$, and deduce
$$1 = \theta_{\mathbb{D}}\, h' U A \bmod h.$$

The value of the zero-test is false, and the result of the inversion is $\theta_{\mathbb{D}}\, h' U \operatorname{rem} h$.

LEMMA 4.1. *Let $f \in \mathbb{K}[x]$ be monic and separable of degree d, and let g and h be non-constant monic polynomials in $\mathbb{K}[x]$ such that $f = gh$. Given the Bézout relation $1 = Af' + Bf$, the Bézout relations for g' and g, and h' and h can be computed by a straight-line program in time $O(\mathsf{M}_{\mathbb{K}}(d))$.*

Proof. From the given Bézout relation we have

$$1 = Agh' + (Bg + Ag')\,h.$$

Since $\deg(((Ag) \operatorname{rem} h)\,h')$ and $\deg(((Bg + Ag') \operatorname{rem} h')\,h)$ are strictly less than $\deg(h'h)$, reduction of this relation modulo $h'h$ yields

$$1 = ((Ag) \operatorname{rem} h)\,h' + ((Bg + Ag') \operatorname{rem} h')\,h.$$

This is the desired Bézout relation for h' and h. By symmetry, the Bézout relation for g' and g can be computed in a similar way. □

4.2. Main complexity result with one algebraic parameter

We are now ready to present the main complexity bound of this section, in terms of the detailed cost functions defined in section 2.1.

THEOREM 4.2. *Let $\mathbb{A} = \mathbb{K}[x]/(\mu_{\mathbb{A}}(x))$ be an explicitly separable extension of $\mathbb{K}$ of degree $d := \deg \mu_{\mathbb{A}} \geqslant 2$, and let T be a computation tree over $\mathbb{K}$-algebras of detailed cost $(\tau_{\text{in}}, \tau_{\text{out}}, \tau_{\text{add}}, \tau_{\text{mul}}, \tau_{\text{div}})$. Then the panoramic evaluation of T over $\mathbb{A}$ by Algorithm 3.2 takes time*

$$O((\tau_{\text{add}}\, d + (\tau_{\text{out}} + \tau_{\text{mul}})\,\mathsf{M}_{\mathbb{K}}(d) + (\tau_{\text{in}} + \tau_{\text{div}})\,\mathsf{M}_{\mathbb{K}}(d)\log d)\log d).$$

Proof. We use Algorithm 3.2 in combination with the technique of delayed reductions from section 3.7. This means that we avoid the cost of conversions, at the price of performing arithmetic operations in subalgebras of $\mathbb{A}$ in the algebra $\mathbb{A}$ itself. More precisely, the cost of one directed evaluation of T over $\mathbb{A}$ is the sum of:

- $O(\tau_{\text{add}}\, d)$ operations in $\mathbb{K}$ for the nodes of type in $\mathbb{K}_0 \cup \mathbb{K}_1 \cup \{+, -\}$,
- $O(\tau_{\text{mul}}\,\mathsf{M}_{\mathbb{K}}(d))$ operations in $\mathbb{K}$ for the nodes of type $\times$,
- $O(\tau_{\text{div}}\,\mathsf{M}_{\mathbb{K}}(d)\log d)$ operations in $\mathbb{K}$ for the directed zero-tests and inversions,
- $O(\tau_{\text{out}}\,\mathsf{M}_{\mathbb{K}}(d))$ operations in $\mathbb{K}$ for the reductions in the output nodes of the path (i.e. the "finalization step" in the terminology of section 3.7).

Let $\mathbb{A} \cong \mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ denote the directed splitting obtained at the end of step 1 of Algorithm 3.2. By construction we have

$$\dim_{\mathbb{K}} \mathbb{H}_i \leqslant d/2 \text{ for all } i = 1, \ldots, \ell. \tag{4.2}$$

Fast multi-remaindering yields $\pi_{\mathbb{A} \to \mathbb{H}_i}(a)$ for $i = 1, \ldots, \ell$ in time $O(\tau_{\text{in}} \mathsf{M}_{\mathbb{K}}(d) \log d)$.

We recursively perform the panoramic evaluation of T over $\mathbb{H}_i$ for all $i = 1, \ldots, \ell$. Let $\mathsf{C}(d)$ be the cost of one panoramic evaluation of T over an algebra of dimension $\leqslant d$. We have shown that there exists a universal constant c such that

$$\mathsf{C}(d) \leqslant c(\tau_{\text{add}}\, d + (\tau_{\text{out}} + \tau_{\text{mul}})\, \mathsf{M}_{\mathbb{K}}(d) + (\tau_{\text{in}} + \tau_{\text{div}})\, \mathsf{M}_{\mathbb{K}}(d) \log d) + \sum_{i=1}^{\ell} \mathsf{C}(\dim_{\mathbb{K}} \mathbb{H}_i).$$

In view of (4.2), we conclude by unrolling this inequality at most $\left\lceil \frac{\log d}{\log 2} \right\rceil$ times. □

Example 4.3. *Continued from Example 3.12.* The first directed evaluation of T performs $O(\kappa\, \mathsf{M}_{\mathbb{K}}(\delta) \log(\kappa\, \delta))$ operations in $\mathbb{K}$. One directed evaluation of T over $\mathbb{Q}[x]/(x^2 - p_i)$ takes time $O(i \log \kappa)$, for $i = 1, \ldots, \kappa$. In this example, the resulting cost of Algorithm 3.2 is

$$O\left(\kappa\, \mathsf{M}_{\mathbb{K}}(\delta) \log(\kappa\, \delta) + \sum_{i=1}^{\kappa} i \log \kappa\right) = O(\kappa\, \mathsf{M}_{\mathbb{K}}(\delta) \log(\kappa\, \delta)),$$

which is better than the bound of Theorem 4.2 because the recursive depth is constant. If $\mathbb{A}$ is a field, then we notice that the usual evaluation of T at α takes time $O(\kappa\, \mathsf{M}_{\mathbb{K}}(\delta) \log \kappa)$.

4.3. Comparison with previous work

Let us now compare the complexity of directed evaluation with the complexities of other known strategies for a single algebraic parameter. One problem with dynamic evaluation is that it is hard to implement in common programming languages without support for parallelism or high level control structures such as continuations [41]. Early implementations of dynamic evaluation were therefore naive [13], the computation tree essentially being reevaluated for every separate case: see subsection 4.3.1. In theory, using SCHEME-style continuations or UNIX-style forking, it is possible to factor out many common computations between the different cases; see [4]. We analyze these approaches from a complexity point of view in section 4.3.2. Besides dynamic evaluation, various more *ad hoc* approaches have also be proposed for computations with a single algebraic parameter; we will discuss them in section 4.3.3.

4.3.1. Naive dynamic evaluation

When using common sequential programming languages, dynamic evaluation is usually implemented as in Algorithm 3.2, with the following two differences:

- Splittings are no longer required to be directed. In other words, at each evaluation step, the splitting $\mathbb{A} \cong \mathbb{D}_v \oplus \mathbb{H}_{v,1} \oplus \cdots \oplus \mathbb{H}_{v,\ell_v}$ at node v of T is not required to satisfy the conditions $2 \dim_{\mathbb{K}} \mathbb{H}_{v,i} \leqslant \dim_{\mathbb{K}} \mathbb{A}$ for $i = 1, \ldots, \ell_v$. Consequently, whenever a zero-test or an inversion triggers a basic splitting for $\mathbb{D}_v$, the branch where we continue the evaluation is chosen non-deterministically. Such evaluations are said to be *undirected* in the sequel.
- Once the undirected evaluation is finished, the projections in step 2 of Algorithm 3.2 are done naively, without fast multi-remaindering.

Adapting the cost analysis of Algorithm 3.2 leads to the complexity bound

$$O((\tau_{\text{add}}\, d + (\tau_{\text{mul}} + \tau_{\text{in}} + \tau_{\text{out}})\, \mathsf{M}_{\mathbb{K}}(d) + \tau_{\text{div}}\, \mathsf{M}_{\mathbb{K}}(d) \log d)\, d)$$

for naive dynamic evaluation, in terms of the detailed cost functions from section 2.1.

The following example shows that the latter bound is tight in the worst case. So naive dynamic evaluation is in general less efficient than fast panoramic evaluation.

Example 4.4. Let us again consider the family of computation trees of Example 2.1, let

$$\mu(x) := (x^2-p_1)\cdots(x^2-p_\delta),$$

with $\delta > \kappa$, $I := (\mu(x))$, $\mathbb{A} := \mathbb{Q}[x]/I$, and let α represent the image of x in $\mathbb{A}$. Then a possible undirected evaluation of T at $a = \alpha$ performs a basic splitting and may select the splitting

$$\mathbb{Q}[x]/(x^2-p_1) \oplus \mathbb{Q}[x]/((x^2-p_2)\cdots(x^2-p_\delta)).$$

So a possible undirected evaluation ends with the latter splitting and output value 1.

A possible undirected evaluation of T over $\mathbb{Q}[x]/((x^2-p_2)\cdots(x^2-p_\delta))$ may end with the splitting

$$\mathbb{Q}[x]/(x^2-p_2) \oplus \mathbb{Q}[x]/((x^2-p_3)\cdots(x^2-p_\delta))$$

and value 2. Doing so for the rest of the evaluation, we may end with the splitting

$$\mathbb{Q}[x]/(x^2-p_1) \oplus \cdots \oplus \mathbb{Q}[x]/(x^2-p_\kappa) \oplus \mathbb{Q}[x]/((x^2-p_{\kappa+1})\cdots(x^2-p_\delta))$$

and the value of T over $\mathbb{Q}[x]/(x^2-p_i)$ is i for $i = 1, \ldots, \kappa$, and is 0 over $\mathbb{Q}[x]/((x^2-p_{\kappa+1})\cdots(x^2-p_\delta))$.

The total execution time therefore grows at least with

$$\sum_{k=1}^{\kappa} k\,\mathsf{M}_{\mathbb{K}}(\delta-k) \;=\; \Omega\left(\sum_{k=1}^{\kappa} k\,(\delta-k)\right) \;=\; \Omega(\kappa^2(\delta-\kappa)).$$

In comparison to Example 4.3, it turns out that dynamic evaluation may be significantly more expensive than our fast panoramic evaluation based on directed evaluation.

4.3.2. Dynamic evaluation using continuations or forking

The naive implementation of dynamic evaluation using reevaluations of the entire computation tree leads to unnecessary recomputations. High-level control structures such as continuations can be used to avoid this kind of recomputations, by resuming the "recomputations" directly at the points where the splittings occurred. UNIX-style forks can be used to the same effect. We will now illustrate the benefit of these approaches on our running Example 2.1, after which we will present another example that remains problematic even with this type of optimizations.

Example 4.5. Let T be a computation tree of the family introduced in Example 2.1, and let us again consider

$$\mu(x) := (x^2-p_1)\cdots(x^2-p_\delta),$$

with $\delta > \kappa$, $I := (\mu(x))$, $\mathbb{A} := \mathbb{Q}[x]/I$, and let α represent the image of x in $\mathbb{A}$. The dynamic evaluation of T at α encounters an inconsistency when testing whether $(x^2-p_1)^\kappa$ is zero modulo $\mu(x)$. One computation continues modulo x^2-p_1, and so returns 1. Another computation continues modulo $(x^2-p_2)\cdots(x^2-p_\delta)$ and a new splitting occurs at the zero-test of $(x^2-p_2)^\kappa$, etc. Counting common computations in branches only once, the total cost of the dynamic evaluation is bounded by

$$O(\kappa\,\mathsf{M}_{\mathbb{K}}(\delta)\log\kappa + \kappa\,\mathsf{M}_{\mathbb{K}}(\delta)\log\delta) = O(\kappa\,\mathsf{M}_{\mathbb{K}}(\delta)\log(\kappa\delta))$$

which is essentially the same as the cost of our fast panoramic evaluation: see Example 4.3.

Example 4.6. Let us now consider the following (still somewhat artificial) program over $\mathbb{Q}$-algebras:

for i from 1 to κ do
  $f_i := (x^2 - p_i)^\kappa$
for i from 1 to κ do
  if $f_i = 0$ then $f_i := f_i + 1$

Evaluated over a field $\mathbb{A}$ of degree d over $\mathbb{K}$, the total cost of this program is

$$\sum_{i=1}^{\kappa} \mathsf{M}_{\mathbb{K}}(d) \log \kappa = \tilde{O}(\kappa d).$$

Let us now take $\mathbb{A} := \mathbb{Q}[x]/I$ with I as in the previous example. The dynamic evaluation executes the first loop sequentially with cost $\sum_{i=1}^{\kappa} \mathsf{M}_{\mathbb{K}}(\delta) \log \kappa = \tilde{O}(\kappa\,\delta)$. Then it splits at the zero-test of $(x^2 - p_1)$ modulo $\mu(x)$ into two branches:

- The first branch works modulo $x^2 - p_1$ and reduces $f_2, \ldots, f_\kappa$ modulo $x^2 - p_1$, which amounts to $O((\kappa - 1)\,\delta)$ operations in $\mathbb{K}$. The costs of the zero-tests totalize $O((\kappa - 1)\,\mathsf{M}_{\mathbb{K}}(2) \log 2)$.
- The second branch works modulo $(x^2 - p_2) \cdots (x^2 - p_\delta)$. The zero-test of f_2 costs $O(\mathsf{M}_{\mathbb{K}}(\delta - 1) \log(\delta - 1))$, and causes the present branch to split up into two:
  - The first branch works modulo $x^2 - p_2$ and reduces $f_3, \ldots, f_\kappa$, which amounts to $O((\kappa - 2)\,\delta)$ operations in $\mathbb{K}$. The costs of the zero-tests totalize $O((\kappa - 2)\,\mathsf{M}_{\mathbb{K}}(2) \log 2)$.
  - The second branch works modulo $(x^2 - p_3) \cdots (x^2 - p_\delta)$. The zero-test of f_3 costs $O(\mathsf{M}_{\mathbb{K}}(\delta - 2) \log(\delta - 2))$, and causes the present branch to split up into two:
    - …

The total execution time when using dynamic evaluation is therefore

$$\Omega\left(\sum_{i=1}^{\kappa} \mathsf{M}_{\mathbb{K}}(\delta) \log \kappa + \sum_{i=1}^{\kappa} \left((\kappa - i)\,\delta + \mathsf{M}_{\mathbb{K}}(\delta - i) \log(\delta - i)\right)\right) = \Omega(\kappa^2\,\delta).$$

By Theorem 4.2, the execution time using fast panoramic evaluation is $\tilde{O}(\kappa\,\delta)$, which is significantly better.

4.3.3. Specific algorithms

The bottleneck of dynamic evaluation in Example 4.6 lies in the projections of data from parametric algebras into smaller ones. It is important to stress that this cost cannot be reduced by executing the branches in a judicious order or by cleverly factoring out common computations in branches using continuations. In our fast panoramic evaluation strategy, this means that fast multi-remaindering plays a crucial role for handling these projections efficiently.

The above examples show that determining the best strategy to compute a panoramic value for a given computation tree is not an obvious problem: in fact, computation trees and more general programs may often be modified in order to accelerate panoramic evaluations. Let us now consider some specific computation trees for which efficient *ad hoc* strategies have been developed.

For panoramic quasi-inversions in $\mathbb{A}$, gcds, and coprime factorizations in $\mathbb{A}[y]$, Dahan *et al.* have designed fast algorithms in [8]. For instance, they showed that a panoramic gcd in $\mathbb{A}[y]_{\leqslant n}$ could be obtained with cost

$$O(\mathsf{M}_{\mathbb{K}}(d)\log d\,\mathsf{M}_{\mathbb{K}}(n)\log n); \tag{4.3}$$

see [8, Propositions 2.4 and 4.1]. For this problem, Theorem 4.2 leads to the cost

$$O((\mathsf{M}_{\mathbb{K}}(n)\,\mathsf{M}_{\mathbb{K}}(d)\log n + n\,\mathsf{M}_{\mathbb{K}}(d)\log d)\log d),$$

which can be further improved to

$$O((\mathsf{M}_{\mathbb{K}}(d\,n)\log n + n\,\mathsf{M}_{\mathbb{K}}(d)\log d)\log d) \tag{4.4}$$

by using Kronecker substitution to multiply polynomials in $(\mathbb{K}[x]/(\mu(x)))[y]_{\leqslant n}$ in time $O(\mathsf{M}_{\mathbb{K}}(d\,n))$. If $\log^2 n = o(\log d)$, then this cost is slightly higher than (4.3). If $d = O(n)$, then (4.4) simplifies to $O(d\,\mathsf{M}_{\mathbb{K}}(n)\log n\log d)$, which is slightly better than (4.3).

Another interesting application of fast panoramic evaluation is the computation of the determinant Δ of an $n\times n$ matrix M with entries in $\mathbb{A} = \mathbb{K}[x]/(\mu(x))$. By means of Berkowitz' algorithm [1], such a determinant can be computed in time $O(n^{\omega+1})$ by a straight-line program over $\mathbb{A}$, and therefore in time $O(n^{\omega+1}\,\mathsf{M}_{\mathbb{K}}(d))$ by a straight-line program over $\mathbb{K}$. This cost decreases to $\tilde{O}(n^{\omega/2+2})\,\mathsf{M}_{\mathbb{K}}(d)$ by using [32], and even to $O(n^{\omega+1/2}\,\mathsf{M}_{\mathbb{K}}(d))$ thanks to [40], whenever inverses of $2,3,\ldots,n$ are given in $\mathbb{K}$. We can do better with fast panoramic evaluation: we first compute the determinant of M over $\mathbb{A}$ as if it were a field by means of a computation tree that performs $O(n^{\omega})$ ring operations and $O(n^2)$ zero-tests and inversions. By Theorem 4.2, this can be done in time

$$O((n^{\omega}\,\mathsf{M}_{\mathbb{K}}(d) + n^2\,\mathsf{M}_{\mathbb{K}}(d)\log d)\log d).$$

The resulting panoramic value corresponds to the residues of Δ modulo several pairwise coprime factors of μ. Chinese remaindering finally allows to recover the actual determinant Δ using $O(\mathsf{M}_{\mathbb{K}}(d)\log d)$ additional operations in $\mathbb{K}$.

5. SEVERAL ALGEBRAIC PARAMETERS

From now on, we focus on the complexity of panoramic evaluation for $t\geqslant 2$ algebraic parameters. Such a tower is determined by algebraic parameters $\alpha_1,\ldots,\alpha_t$, where α_1 is constrained by an algebraic equation over $\mathbb{K}$, and α_i is constrained by an algebraic equation over $\mathbb{K}[\alpha_1,\ldots,\alpha_{i-1}]$ for $i=2,\ldots,t$. It is natural to apply the method of the previous sections for a single algebraic parameter in a recursive manner. But we have to face a new difficulty: directed zero-tests and inversions in $\mathbb{K}[\alpha_1,\ldots,\alpha_t]$ will involve occasional zero-tests and inversions in $\mathbb{K}[\alpha_1,\ldots,\alpha_{t-1}]$, that will themselves involve occasional zero-tests and inversions in $\mathbb{K}[\alpha_1,\ldots,\alpha_{t-2}]$, etc. The first goal of this section is to design data structures that allow us to create new branches for free during directed evaluation. At the end of a directed evaluation we need to project input data into all the residual branches. Performing this task efficiently is the second goal of this section.

Data structures play an important role in the efficiency of operations required by parametric frameworks. From now on, we will only manipulate parametric algebras whose defining ideals are generated by triangular sets. In other words, all parametric algebras will be represented by towers $(\mathbb{A}_i)_{i\leqslant t}$ over $\mathbb{A}_0 = \mathbb{K}$, as in section 1.1. We recall that $(\mathbb{A}_i)_{i\leqslant t}$ is also assumed to be absolutely reduced, i.e. $\bar{\mathbb{K}} \otimes \mathbb{A}_t$ is a product of fields. A tower $(\mathbb{A}_i)_{i\leqslant t}$ is said to be *explicitly separable* if we are given the Bézout cofactors θ_i and ξ_i in $\mathbb{A}_{i-1}[x_i]$ such that $1 = \theta_i \mu_i' + \xi_i \mu_i$ for $i = 1, \ldots, t$.

In order to simplify complexity analyses, it is convenient to assume that $d_i := \deg \mu_i \geqslant 2$ for $i = 1, \ldots, t$. In particular, $d = d_1 \cdots d_t \geqslant 2$. This restriction is harmless, since extensions of degree $d_i = 1$ can be discarded without cost in our model of computation trees.

5.1. Tower factorizations

Consider two towers $(\mathbb{A}_i)_{i\leqslant t}$ and $(\mathbb{B}_i)_{i\leqslant t}$ of the same height t and over the same base field $\mathbb{K}$. Let $(\mu_i)_{i\leqslant t}$ and $(\nu_i)_{i\leqslant t}$ denote the respective defining polynomials of $(\mathbb{A}_i)_{i\leqslant t}$ and $(\mathbb{B}_i)_{i\leqslant t}$. We say that $(\mathbb{B}_i)_{i\leqslant t}$ is a *factor tower* of $(\mathbb{A}_i)_{i\leqslant t}$ if $Z_{\mathbb{B}_t} \subseteq Z_{\mathbb{A}_t}$. This is the case if, and only if, there exist natural projections $\pi_{\mathbb{A}_i \to \mathbb{B}_i}$ (that naturally extend to projections $\pi_{\mathbb{A}_i \to \mathbb{B}_i}$: $\mathbb{A}_i[x_i] \to \mathbb{B}_i[x_i]$ in a coefficient-wise manner) such that:

- ν_i divides $\pi_{\mathbb{A}_{i-1} \to \mathbb{B}_{i-1}}(\mu_i)$ for $i = 1, \ldots, t$.
- $\pi_{\mathbb{A}_i \to \mathbb{B}_i}$ sends an element $a \in \mathbb{A}_i$ represented by $A(x_i) = \sum_{k=0}^{d_i - 1} A_k x_i^k \in \mathbb{A}_{i-1}[x_i]_{<d_i}$ to

$$\pi_{\mathbb{A}_{i-1} \to \mathbb{B}_{i-1}}(A(x_i)) = \sum_{k=0}^{d_i - 1} \pi_{\mathbb{A}_{i-1} \to \mathbb{B}_{i-1}}(A_k)\, x_i^k \operatorname{rem} \nu_i(x_i),$$

  for $i = 1, \ldots, t$.

Given such a factor tower $(\mathbb{B}_i)_{i\leqslant t}$ of $(\mathbb{A}_i)_{i\leqslant t}$, let us now study the cost $\mathsf{C}(\mathbb{A}_i \to \mathbb{B}_i)$ of computing one projection $\pi_{\mathbb{A}_i \to \mathbb{B}_i}(x) \in \mathbb{B}_i$ of an element $x \in \mathbb{A}_i$ with $i \leqslant t$.

LEMMA 5.1. *Let $(\mathbb{B}_i)_{i\leqslant t}$ be a factor tower of $(\mathbb{A}_i)_{i\leqslant t}$. The projection $\pi_{\mathbb{A}_t \to \mathbb{B}_t}(a)$ of an element $a \in \mathbb{A}_t$ can be computed by a straight-line program in time*

$$\mathsf{C}(\mathbb{A}_t \to \mathbb{B}_t) = O\left(\sum_{i=1}^{t} \mathsf{M}_{\mathbb{B}_{i-1}}(d_i)\, d_{i+1} \cdots d_t\right) = O(C^t \mathsf{M}_{\mathbb{K}}(d)).$$

Proof. Let $A \in \mathbb{A}_{t-1}[x_t]_{<d_t}$ denote the preimage of a. We recursively apply $\pi_{\mathbb{A}_{t-1} \to \mathbb{B}_{t-1}}$ to the coefficients of A. Then, the reduction of $\pi_{\mathbb{A}_{t-1} \to \mathbb{B}_{t-1}}(A)$ modulo ν_t can be done by a straight-line program with cost $O(\mathsf{M}_{\mathbb{B}_{t-1}}(d_t))$. Consequently, there exists a universal constant c such that the cost function $\mathsf{C}(\mathbb{A}_t \to \mathbb{B}_t)$ satisfies

$$\mathsf{C}(\mathbb{A}_t \to \mathbb{B}_t) \;\leqslant\; c\, \mathsf{M}_{\mathbb{B}_{t-1}}(d_t) + \mathsf{C}(\mathbb{A}_{t-1} \to \mathbb{B}_{t-1})\, d_t.$$

Unrolling this inequality yields

$$\mathsf{C}(\mathbb{A}_t \to \mathbb{B}_t) = O\left(\sum_{i=1}^{t} \mathsf{M}_{\mathbb{B}_{i-1}}(d_i)\, d_{i+1} \cdots d_t\right).$$

Taking $\mathsf{M}_{\mathbb{B}_{i-1}}(d_i) = O(C^{i-1} \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i))$ from Proposition 2.6 concludes the proof. □

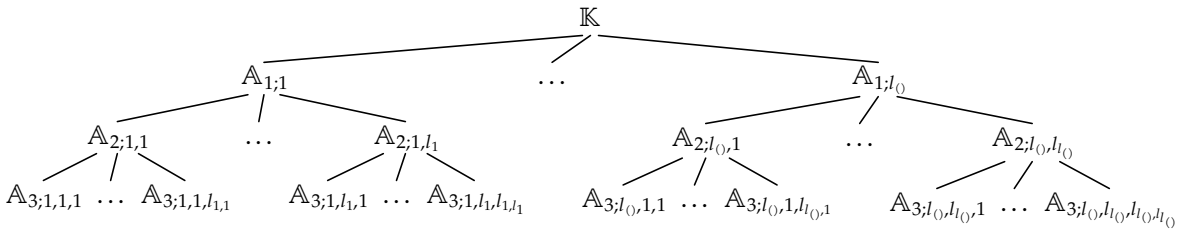


Figure 5.1. Example of a tree factorization of a tower of height $t=3$.

In the sequel, "()" stands for the empty tuple. Let Σ_k be an index set of k-tuples of integers with $\Sigma_0=\{()\}$ and

$$\Sigma_k=\{(\sigma_1,\ldots,\sigma_{k-1},i):(\sigma_1,\ldots,\sigma_{k-1})\in\Sigma_{k-1},i\in\{1,\ldots,l_{\sigma_1,\ldots,\sigma_{k-1}}\}\},$$

for integers $l_{\sigma_1,\ldots,\sigma_{k-1}}$ and $k=1,\ldots,t$. Consider a family $((\mathbb{A}_{i;\sigma})_{i\leqslant t})_{\sigma\in\Sigma_t}$ of factor towers of $(\mathbb{A}_i)_{i\leqslant t}$ with the property that $\mathbb{A}_{k;\sigma_1,\ldots,\sigma_t}$ only depends on $\sigma_1,\ldots,\sigma_k$ for all $(\sigma_1,\ldots,\sigma_t)\in\Sigma_t$, and write $\mathbb{A}_{k;\sigma_1,\ldots,\sigma_k}:=\mathbb{A}_{k;\sigma_1,\ldots,\sigma_t}$. We say that such a family of factors forms a *tree factorization* of $(\mathbb{A}_i)_{i\leqslant t}$ if the variety $Z_{\mathbb{A}_t}$ is partitioned into $Z_{\mathbb{A}_t}=\coprod_{\sigma\in\Sigma_t}Z_{\mathbb{A}_{t;\sigma}}$. If $k>0$ and $\sigma\in\Sigma_k$, then we also write $\mu_{k;\sigma}\in\mathbb{A}_{k;\sigma_1,\ldots,\sigma_{k-1}}[x_k]$ for the defining polynomial of $\mathbb{A}_{k;\sigma}$ over $\mathbb{A}_{k-1;\sigma}$.

It is convenient to represent such a factorization by a labeled tree (see Figure 5.1): the nodes are identified with the index set made of the disjoint union

$$\Sigma_0\amalg\Sigma_1\amalg\ldots\amalg\Sigma_t,$$

and each node $\sigma\in\Sigma_k$ is labeled with the algebraic extension $\mathbb{A}_{k;\sigma}$. The parent of the node $\sigma\in\Sigma_k$ with $k>0$ is simply the node $(\sigma_1,\ldots,\sigma_{k-1})\in\Sigma_{k-1}$. Each individual factor tower $(\mathbb{A}_{i;\sigma})_{i\leqslant t}$ corresponds to a path from the root to a leaf.

Given $k\leqslant t$ and $\sigma\in\Sigma_k$, let

$$\Sigma_{t;\sigma}:=\{\tau:(\sigma,\tau)\in\Sigma_t\}.$$

Projecting the equality

$$Z_{\mathbb{A}_t}=\coprod_{\sigma\in\Sigma_k}\coprod_{\tau\in\Sigma_{t;\sigma}}Z_{\mathbb{A}_{t;\sigma,\tau}}$$

on the first k coordinates, we observe that the projection of $Z_{\mathbb{A}_{t;\sigma,\tau}}$, for given $\sigma\in\Sigma_k$, is the same for all $\tau\in\Sigma_{t;\sigma}$, and equals $Z_{\mathbb{A}_{k;\sigma}}$, whence $Z_{\mathbb{A}_k}=\coprod_{\sigma\in\Sigma_k}Z_{\mathbb{A}_{k;\sigma}}$. Consequently, $((\mathbb{A}_{i;\sigma_1,\ldots,\sigma_i})_{i\leqslant k})_{\sigma\in\Sigma_k}$ forms a tree factorization of the sub-tower $(\mathbb{A}_i)_{i\leqslant k}$. From an algebraic point of view, this means that we have a natural isomorphism

$$\mathbb{A}_k\ \cong\ \bigoplus_{\sigma\in\Sigma_k}\mathbb{A}_{k;\sigma}.$$

5.2. Directed evaluation in the multivariate case

Consider an explicitly separable tower $(\mathbb{A}_i)_{i\leqslant t}$ and assume that we wish to compute the panoramic evaluation of a computation tree T over $\mathbb{A}_t$. In order to apply Algorithm 3.2 in this multivariate context, it would be natural to specify a parametric framework, as we did in the univariate case. Since we will exclusively focus on the directed approach from now on, it turns out to be simpler and more straightforward to describe how to perform the directed evaluation and the projections in steps 1 and 2.

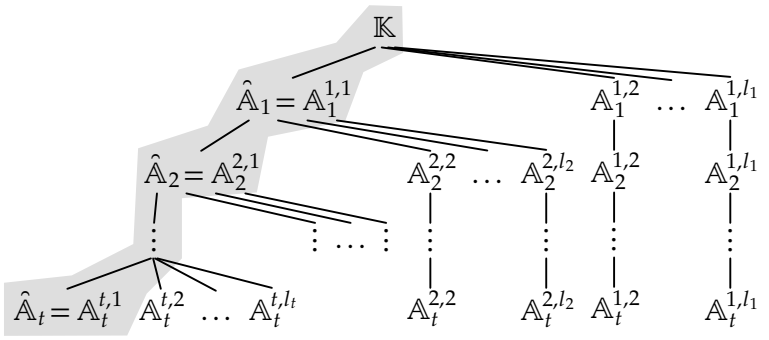


Figure 5.2. Illustration of a generic tower factorization involved in a directed evaluation. The principal branch is highlighted.

Informally speaking, we want to work with respect to a "factor of highest possible degree" at each level. This leads to special types of factorizations of $(\mathbb{A}_i)_{i\leqslant t}$ that are illustrated in Figure 5.2. The actual computations are done in the algebras from the highlighted "principal branch". The other "residual branches" are dealt with at a next iteration, when the computations for the principal branch have been completed.

Technically speaking, a *directed splitting* of $\mathbb{A}_t$ will be represented by:

- Vectors $\vec{\mu}_i := (\mu_{i,1}, \ldots, \mu_{i,l_i})$ of polynomials in $\mathbb{A}_{i-1}[x_i]$ for $i = 1, \ldots, t$, such that:
  - $\hat{\mu}_1 := \mu_{1,1}, \ldots, \hat{\mu}_t := \mu_{t,1}$ are the defining polynomials of a factor tower of $(\mathbb{A}_i)_{i\leqslant t}$, written $(\hat{\mathbb{A}}_i)_{i\leqslant t}$, and called the *principal branch*.
  - $\pi_{\mathbb{A}_{i-1}\to\hat{\mathbb{A}}_{i-1}}(\mu_i) = \hat{\mu}_i\, \pi_{\mathbb{A}_{i-1}\to\hat{\mathbb{A}}_{i-1}}(\mu_{i,2}) \cdots \pi_{\mathbb{A}_{i-1}\to\hat{\mathbb{A}}_{i-1}}(\mu_{i,l_i})$ for $i = 1, \ldots, t$.
  - $2 \deg \mu_{i,k} \leqslant \deg \mu_i$ for $k = 2, \ldots, l_i$.
- The Bézout relations $1 = \hat{\theta}_i \hat{\mu}_i' + \hat{\xi}_i \hat{\mu}_i$ for $i = 1, \ldots, t$, so $(\hat{\mathbb{A}}_i)_{i\leqslant t}$ is explicitly separable.

Each factor $\pi_{\mathbb{A}_{j-1}\to\hat{\mathbb{A}}_{j-1}}(\mu_{j,k})$ with $j = 1, \ldots, t$ and $k = 2, \ldots, l_j$ gives rise to a factor tower $(\mathbb{A}_i^{j,k})_{i\leqslant t}$ of $(\mathbb{A}_i)_{i\leqslant t}$, called a *residual branch*, as follows:

- For $i < j$, we set $\mathbb{A}_i^{j,k} := \hat{\mathbb{A}}_i$.
- $\mathbb{A}_j^{j,k} := \hat{\mathbb{A}}_{j-1}[x_j] / (\pi_{\mathbb{A}_{j-1}\to\hat{\mathbb{A}}_{j-1}}(\mu_{j,k}(x_j)))$.
- For $i > j$, the defining polynomial of $\mathbb{A}_i^{j,k}$ over $\mathbb{A}_{i-1}^{j,k}$ is the projection of $\mu_i(x_i)$ in $\mathbb{A}_{i-1}^{j,k}[x_i]$.

The tree representation of the resulting tower factorization is summarized in Figure 5.2. The actual directed splitting of $\mathbb{A}_t$ represented in this manner is

$$\mathbb{A}_t \;\cong\; \hat{\mathbb{A}}_t \oplus \mathbb{A}_t^{t,2} \oplus \cdots \oplus \mathbb{A}_t^{t,l_t} \oplus \cdots \oplus \mathbb{A}_t^{2,2} \oplus \cdots \oplus \mathbb{A}_t^{2,l_2} \oplus \cdots \oplus \mathbb{A}_t^{1,2} \oplus \cdots \oplus \mathbb{A}_t^{1,l_1}.$$

Notice that a directed splitting $\vec{\mu}_1, \ldots, \vec{\mu}_t$ of $(\mathbb{A}_i)_{i\leqslant t}$ naturally induces a directed splitting $\vec{\mu}_1, \ldots, \vec{\mu}_j$ of $(\mathbb{A}_i)_{i\leqslant j}$ for $j = 1, \ldots, t$.

Now the main idea is to perform the directed evaluation of a computation tree over the principal branch, while postponing evaluations over residual branches. With our representation of directed splittings, creating residual branches is essentially free of charge; the corresponding tower representations will only be computed after completion of the entire directed evaluation, using a dedicated "finalization" procedure that will be detailed in subsection 5.4 below. After the finalization, the projections from step 2 of Algorithm 3.2 can be carried out efficiently using fast multi-remaindering; this will also be detailed in subsection 5.4.

As in the univariate case, it will be convenient to use the technique of delayed reductions from section 3.7. During directed evaluations, this means that we will actually represent elements in $\hat{\mathbb{A}}_i$ by elements in $\mathbb{A}_i$, for $i = 1, \ldots, t$. In particular, operations in $\mathbb{K}_0 \cup \mathbb{K}_1 \cup \{+, -, \times\}$ on elements in $\hat{\mathbb{A}}_i$ are really performed in $\mathbb{A}_i$. We recall that the main benefit of this representation is that projections $\pi_{\hat{\mathbb{A}}_i \to \check{\mathbb{A}}_i}$ are free of charge, whenever $\check{\mathbb{A}}_i$ is a quotient algebra of $\hat{\mathbb{A}}_i$ whose elements are also redundantly represented by elements in $\mathbb{A}_i$. On the other hand, we will resort to reduced representations for directed zero-tests and inversions. We will also systematically reduce all output values. Notice that elements in $\hat{\mathbb{A}}_i$ can be reduced in time $O(C^i \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i))$, by Lemma 5.1. In the next subsection, we detail how to perform directed zero-tests and inversions.

5.3. Directed zero-tests and inversions

In the case of a single algebraic parameter, we reduced directed inversions and zero-tests to extended gcd computations. With several parameters the problem is more difficult. An element a in $\mathbb{A}_t$ can be tested to be invertible by means of a straight-line program over $\mathbb{K}$ with polynomial cost and a single zero-test in $\mathbb{K}$: it essentially suffices to compute the determinant of the multiplication endomorphism $\mathbb{A}_t \ni x \mapsto a\,x \in \mathbb{A}_t$ using Berkowitz' algorithm [1]. But the cost of this approach is more than quadratic in d, which is not satisfactory for our purposes. We now develop a more efficient strategy based on recursive calls of the fast extended gcd algorithm in the directed evaluation model. Note that a similar recursive approach was previously used in the contexts of dynamic evaluation and triangular sets.

Algorithm 5.1

Input. An explicitly separable tower $(\mathbb{A}_i)_{i \leqslant t}$, a directed splitting $(\overrightarrow{\mu_i})_{i \leqslant t}$ of $(\mathbb{A}_i)_{i \leqslant t}$, and $a \in \hat{\mathbb{A}}_t$, where $(\hat{\mathbb{A}}_i)_{i \leqslant t}$ denotes the principal branch of $(\overrightarrow{\mu_i})_{i \leqslant t}$.

Output. A directed splitting $(\overleftarrow{\mu_i})_{i \leqslant t}$ of $(\mathbb{A}_i)_{i \leqslant t}$, with principal branch $(\check{\mathbb{A}}_i)_{i \leqslant t}$, the boolean value of the zero-test of $\check{a} := \pi_{\hat{\mathbb{A}}_t \to \check{\mathbb{A}}_t}(a)$, where $\check{a}$ is either zero or invertible in $\check{\mathbb{A}}_t$, and the inverse $\check{a}^{-1}$ whenever $\check{a}$ is non-zero.

1. If $t = 0$ then return true if $a = 0$ or false and a^{-1} otherwise. The directed splitting is left unchanged.
2. Let $A \in \hat{\mathbb{A}}_{t-1}[x_t]_{<d_t}$ denote the preimage of a, and recursively compute the extended monic gcd of A and $\hat{\mu}_t$ over $\hat{\mathbb{A}}_{t-1}$, using directed evaluation of a computation tree for extended gcds, with the directed splitting $(\overrightarrow{\mu_i})_{i \leqslant t-1}$ as extra input.

   Let $(\overleftarrow{\mu_i})_{i \leqslant t-1}$ denote the directed splitting obtained in return and let $(\check{\mathbb{A}}_i)_{i \leqslant t-1}$ be its principal branch. This extended gcd computation also returns a monic polynomial $g \in \check{\mathbb{A}}_{t-1}[x_t]$, and U, V in $\check{\mathbb{A}}_{t-1}[x_t]$ such that the following Bézout relation holds:

$$g = U\check{A} + V\check{\mu}_t,$$

   where $\check{A} := \pi_{\hat{\mathbb{A}}_{t-1} \to \check{\mathbb{A}}_{t-1}}(A)$ and $\check{\mu}_t := \pi_{\hat{\mathbb{A}}_{t-1} \to \check{\mathbb{A}}_{t-1}}(\hat{\mu}_t)$.
3. Let $\check{\theta}_t := \pi_{\hat{\mathbb{A}}_{t-1} \to \check{\mathbb{A}}_{t-1}}(\hat{\theta}_t)$, $\check{\xi}_t := \pi_{\hat{\mathbb{A}}_{t-1} \to \check{\mathbb{A}}_{t-1}}(\hat{\xi}_t)$, so that the Bézout relation $1 = \check{\theta}_t\,\check{\mu}'_t + \check{\xi}_t\,\check{\mu}_t$ holds. Compute the reduced representations of $\check{A}$, $\check{\mu}_t$, $\check{\theta}_t$, $\check{\xi}_t$, and g (recall that we delayed these reductions), and then compute $h := \check{\mu}_t / g$.
4. Deduce the Bézout relations $1 = \theta_g g' + \xi_g g$ and $1 = \theta_h h' + \xi_h h$ by means of Lemma 4.1.
5. If $g = 0$, then return true, along with the directed splitting $(\overleftarrow{\mu_i})_{i \leqslant t-1}$, $(\check{\mu}_t, \mu_{t,2}, \ldots, \mu_{t,l_t})$, and the Bézout relation of $\check{\mu}'_t$ and $\check{\mu}_t$.

6. If $g = 1$, then return false, along with the directed splitting $(\overleftarrow{\mu_i})_{i\leqslant t-1}, (\breve{\mu}_t, \mu_{t,2}, \ldots, \mu_{t,l_t})$, the Bézout relation of $\breve{\mu}'_t$ and $\breve{\mu}_t$, and the inverse $U(x_t) \bmod \breve{\mu}_t$ of $\pi_{\hat{\mathbb{A}}_t \to \breve{\mathbb{A}}_t}(a)$.
7. If $2 \deg h \leqslant \deg \mu_t$, then return true, along with the directed splitting $(\overleftarrow{\mu_i})_{i\leqslant t-1}$, $(g, h, \mu_{t,2}, \ldots, \mu_{t,l_t})$, and the Bézout relation of g' and g.
8. Otherwise return false, along with the directed splitting $(\overleftarrow{\mu_i})_{i\leqslant t-1}, (h, g, \mu_{t,2}, \ldots, \mu_{t,l_t})$, the Bézout relation of h' and h, and the inverse $\breve{\theta}_t U h' \bmod h$ of $\pi_{\hat{\mathbb{A}}_t \to \breve{\mathbb{A}}_t}(a)$.

Until the end of the paper, $\mathsf{M}_{\mathbb{K}}(d_1, \ldots, d_i; n)$ represents a cost function for products in $\hat{\mathbb{A}}_i[x]_{<n}$ where $(\hat{\mathbb{A}}_i)_{i\leqslant t}$ can be any principal branch encountered in the directed evaluation model over $(\mathbb{A}_i)_{i\leqslant t}$. Similarly, $\mathsf{P}_{\mathbb{K}}(d_1,\ldots,d_i)$ is a cost function for reduced projections from $\mathbb{A}_i$ onto $\hat{\mathbb{A}}_i$.

PROPOSITION 5.2. *Algorithm 5.1 is correct and takes $O(C^t \mathsf{M}_{\mathbb{K}}(d) \log \bar{d})$ operations in $\mathbb{K}$, where $\bar{d} := \max(d_1, \ldots, d_t)$.*

Proof. In step 8 we have

$$2 \deg g = 2(\deg \mu_{t,1} - \deg h) \leqslant 2(\deg \mu_t - \deg h) \leqslant \deg \mu_t.$$

Consequently, the splittings in the output are directed.

In step 8, the fact that $\breve{\theta}_t U h'$ taken modulo h is the inverse of $\breve{a}$ in $\breve{\mathbb{A}}_t$ can be verified as follows, in the same way as in section 4.1: the Bézout relation of $\breve{A}$ and $\breve{\mu}_t$ gives

$$\breve{\theta}_t h' U \breve{A} \bmod h = \breve{\theta}_t h' g \bmod h,$$

which simplifies to $1 \bmod h$ thanks to the Bézout relation of $\breve{\mu}'_t$ and $\breve{\mu}_t$. The correctness of the algorithm follows from these facts.

Let us write $\mathsf{D}_{\mathbb{K}}(d_1, \ldots, d_t)$ for the cost of the algorithm and recall that we are using the technique of delayed reductions. If $t = 0$ then the cost is $O(1)$. Otherwise, when computing extended gcds using the fast algorithm from section 2.2, step 2 takes

$$O(\mathsf{M}_{\mathbb{K}}(d_1, \ldots, d_{t-1}; d_t) \log d_t + \mathsf{P}_{\mathbb{K}}(d_1, \ldots, d_{t-1})\, d_t) + \mathsf{D}_{\mathbb{K}}(d_1, \ldots, d_{t-1})\, d_t$$

operations in $\mathbb{K}$. Step 3 takes

$$O(\mathsf{P}_{\mathbb{K}}(d_1, \ldots, d_{t-1})\, d_t + \mathsf{M}_{\mathbb{K}}(d_1, \ldots, d_{t-1}; d_t))$$

further operations, and step 4 takes time $O(\mathsf{M}_{\mathbb{K}}(d_1,\ldots,d_{t-1};d_t))$, by Lemma 4.1. Steps 5 to 8 require $O(\mathsf{M}_{\mathbb{K}}(d_1 \cdots d_{t-1}; d_t))$ operations in $\mathbb{K}$. Therefore, there exists a universal constant c_0 such that

$$\mathsf{D}_{\mathbb{K}}(d_1,\ldots,d_t) \leqslant c_0 (\mathsf{M}_{\mathbb{K}}(d_1,\ldots,d_{t-1};d_t) \log d_t + \mathsf{P}_{\mathbb{K}}(d_1,\ldots,d_{t-1})\, d_t) + \mathsf{D}_{\mathbb{K}}(d_1,\ldots,d_{t-1})\, d_t. \quad (5.1)$$

Proposition 2.6 and Lemma 5.1 lead to

$$\begin{aligned} \mathsf{M}_{\mathbb{K}}(d_1, \ldots, d_{t-1}; d_t) &= O(C^{t-1} \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_t)) \\ \mathsf{P}_{\mathbb{K}}(d_1, \ldots, d_{t-1}) &= O(C^{t-1} \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_{t-1})), \end{aligned}$$

whence

$$\mathsf{D}_{\mathbb{K}}(d_1, \ldots, d_t) \leqslant c_1 C^{t-1} \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_t) \log d_t + \mathsf{D}_{\mathbb{K}}(d_1, \ldots, d_{t-1})\, d_t,$$

for some universal constant c_1. Using $C > 1$, unrolling the latter inequality leads to the claimed bound. □

5.4. Finalizing the residual branches

As before, let $(\mathbb{A}_i)_{i\leqslant t}$ be a separable tower with defining polynomials μ_i of degree $d_i \geqslant 2$, and denote $d := d_1 \cdots d_t$. At the end of a directed evaluation, we must recover the tower factorization of $(\mathbb{A}_i)_{i\leqslant t}$ induced by the directed splitting. We also need efficient routines for projecting onto the residual branches. In this subsection (and contrary to the previous subsection), we use the traditional, reduced representations for elements in towers. We begin with a simple lemma for the computation of Bézout relations induced by factorizations.

LEMMA 5.3. *Let $f \in \mathbb{K}[x]$ be monic and separable of degree d, and let $g_1, \ldots, g_s$ be non constant monic polynomials in $\mathbb{K}[x]$ such that $f = g_1 \cdots g_s$. Given the Bézout relation $1 = Af' + Bf$ of f' and f, the Bézout relations of g_i' and g_i for $i = 1, \ldots, s$ can be computed by a straight-line program in time $O(\mathsf{M}_{\mathbb{K}}(d) \log s)$.*

Proof. We first construct the subproduct tree of $g_1, \ldots, g_s$: it is defined as the set of the sub-products defined recursively as follows:

- the set $\{g_1\}$ if $s = 1$,
- the union of $\{g_1 \cdots g_s\}$ and the subproduct trees of $g_1, \ldots, g_{\lceil s/2 \rceil}$ and $g_{\lceil s/2 \rceil + 1}, \ldots, g_s$ if $s > 1$.

A straightforward divide and conquer algorithm computes all these products with $O(\mathsf{M}_{\mathbb{K}}(d) \log s)$ arithmetic operations in $\mathbb{K}$. Then, we obtain the Bézout relations for $(g_1 \cdots g_{\lceil s/2 \rceil})'$ and $g_1 \cdots g_{\lceil s/2 \rceil}$, as well as for $(g_{\lceil s/2 \rceil + 1} \cdots g_s)'$ and $g_{\lceil s/2 \rceil + 1} \cdots g_s$, in time $O(\mathsf{M}_{\mathbb{K}}(d))$ by Lemma 4.1. We conclude using a standard induction argument on s. □

LEMMA 5.4. *Given a directed splitting $(\vec{\mu_i})_{i\leqslant t}$ as above, the explicitly separable factor towers $(\mathbb{A}_i^{j,k})_{i\leqslant t}$ for $j = 1, \ldots, t$ and $k = 1, \ldots, l_j$ can be computed in time $O(C^t \mathsf{M}_{\mathbb{K}}(d) \log \bar{d})$ by a straight-line program, where $\bar{d} := \max(d_1, \ldots, d_t)$. In addition, both directions of the isomorphism*

$$\mathbb{A}_t \cong \hat{\mathbb{A}}_t \oplus \bigoplus_{j=1}^{t} \bigoplus_{k=2}^{l_j} \mathbb{A}_t^{j,k} \tag{5.2}$$

can be computed by straight-line programs in time $O(C^t \mathsf{M}_{\mathbb{K}}(d) \log \bar{d})$.

Proof. Until the end of the proof, it is convenient to keep in mind the notation for the tree factorization in Figure 5.2, associated to the directed splitting. Let us write:

- $\mathsf{F}_{\mathbb{K}}(d_1, \ldots, d_t)$ for the cost of the construction of the explicitly separable factor towers $(\mathbb{A}_i^{j,k})_{i\leqslant t}$;
- $\mathsf{Q}_{\mathbb{K}}(d_1, \ldots, d_t)$ for the cost of the projections from $\mathbb{A}_t$ onto $\hat{\mathbb{A}}_t \oplus \bigoplus_{j=1}^{t} \bigoplus_{k=2}^{l_j} \mathbb{A}_t^{j,k}$;
- $\mathsf{R}_{\mathbb{K}}(d_1, \ldots, d_t)$ for the cost of the Chinese remaindering from $\hat{\mathbb{A}}_t \oplus \bigoplus_{j=1}^{t} \bigoplus_{k=2}^{l_j} \mathbb{A}_t^{j,k}$ to $\mathbb{A}_t$.

The proof is done by induction on t. If $t = 0$, then the costs are zero. Assume now that $t \geqslant 1$ and that explicitly separable factor towers $(\mathbb{A}_i^{j,k})_{i\leqslant t-1}$ have been constructed. We thus have the following effective isomorphism:

$$\mathbb{A}_{t-1} \;\cong\; \mathbb{B}_{t-1} \;:=\; \hat{\mathbb{A}}_{t-1} \oplus \bigoplus_{j=1}^{t-1} \bigoplus_{k=2}^{l_j} \mathbb{A}_{t-1}^{j,k}. \tag{5.3}$$

We first project $\mu_t(x_t)$ and the Bézout relation $1 = \theta_t(x_t)\, \mu_t'(x_t) + \xi_t(x_t)\, \mu_t(x_t)$ onto $\hat{\mathbb{A}}_{t-1}[x_t]$ and $\mathbb{A}_{t-1}^{j,k}[x_t]$ for $j = 1, \ldots, t-1$ and $k = 2, \ldots, l_j$ with cost $O(\mathsf{Q}_{\mathbb{K}}(d_1, \ldots, d_{t-1})\, d_t)$. This already gives the defining polynomials of $\mathbb{A}_t^{j,k}$ over $\mathbb{A}_{t-1}^{j,k}$ for $j = 1, \ldots, t-1$ and $k = 2, \ldots, l_j$, along with the corresponding Bézout relations.

We next project the polynomials of $\vec{\mu_t}$ onto $\hat{\mathbb{A}}_{t-1}[x_t]$, namely

$$\hat{\mu}_{t,k} := \pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\mu_{t,k}),$$

for $k=2,\ldots,l_t$, using $O(\mathsf{Q}_{\mathbb{K}}(d_1,\ldots,d_{t-1})\,d_t)$ operations in $\mathbb{K}$ (it is better to use Lemma 5.1 in practice, but the present bound in terms of $\mathsf{Q}_{\mathbb{K}}$ simplifies the analysis). From the already computed $\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\mu_t)$, $\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\theta_t)$, and $\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\xi_t)$, we deduce the Bézout relations of $\hat{\mu}'_{t,k}$ and $\hat{\mu}_{t,k}$ for $k=2,\ldots,l_t$, by means of Lemma 5.3, in time

$$O(\mathsf{M}_{\hat{\mathbb{A}}_{t-1}}(d_t)\log d_t).$$

Overall, we have shown that there exists a universal constant c_1 such that

$$\mathsf{F}_{\mathbb{K}}(d_1,\ldots,d_t) \leqslant c_1\,(\mathsf{M}_{\mathbb{K}}(d_1,\ldots,d_{t-1};d_t)\log d_t + \mathsf{Q}_{\mathbb{K}}(d_1,\ldots,d_{t-1})\,d_t) + \mathsf{F}_{\mathbb{K}}(d_1,\ldots,d_{t-1}). \quad (5.4)$$

Let us now consider both directions of the isomorphism (5.2). Given $a\in\mathbb{A}_t$, represented by $A\in\mathbb{A}_{t-1}[x_t]_{<d_t}$, we compute the projections $\hat{A} := \pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(A)$ and $\pi_{\mathbb{A}_{t-1}\to\mathbb{A}_{t-1}^{j,k}}(A)$ for $j=1,\ldots,t-1$ and $k=2,\ldots,l_j$ in time $O(\mathsf{Q}_{\mathbb{K}}(d_1,\ldots,d_{t-1})\,d_t)$; this already gives the images of a into $\mathbb{A}_t^{j,k}$ for $j=1,\ldots,t-1$ and $k=2,\ldots,l_j$. Then we compute the remainders of $\hat{A}$ modulo $\hat{\mu}_{t,k}$ for $k=1,\ldots,l_t$ in time $O(\mathsf{M}_{\hat{\mathbb{A}}_{t-1}}(d_t)\log d_t)$. So there exists a universal constant c_2 such that

$$\mathsf{Q}_{\mathbb{K}}(d_1,\ldots,d_t) \;\leqslant\; c_2\,\mathsf{M}_{\mathbb{K}}(d_1,\ldots,d_{t-1};d_t)\log d_t + \mathsf{Q}_{\mathbb{K}}(d_1,\ldots,d_{t-1})\,d_t. \quad (5.5)$$

For the inverse Chinese remaindering problem, we are given $a_{t,k}\in\mathbb{A}_t^{t,k}$ for $k=1,\ldots,l_t$, and $a_{j,k}\in\mathbb{A}_t^{j,k}$ for $j=1,\ldots,t-1$ and $k=2,\ldots,l_j$. We have at our disposal $\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\mu_t)$, its factors $\hat{\mu}_{t,k}$ for $k=1,\ldots,l_t$ and the Bézout relation of $\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\mu_t)'$ and $\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\mu_t)$. We compute $a_{t-1,1}\in\hat{\mathbb{A}}_{t-1}[x_t]/(\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\mu_t))$ whose projections are $a_{t,k}$ for $k=1,\ldots,l_t$:

$$A_{t-1,1} = \sum_{k=1}^{t} ((A_{t,k}\,\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\theta_t)\,\hat{\mu}'_{t,k}) \operatorname{rem} \hat{\mu}_{t,k})\,\frac{\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\mu_t)}{\hat{\mu}_{t,k}},$$

where $A_{t-1,1}$ (resp. $A_{t,k}$) is the canonical preimage of $a_{t-1,1}$ (resp. of $a_{t,k}$). Computing $A_{t-1,1}$ costs $O(\mathsf{M}_{\hat{\mathbb{A}}_{t-1}}(d_t)\log d_t)$; see [21, Algorithm 10.20]. The isomorphism (5.3) gives rise to the isomorphism

$$\begin{aligned}
\mathbb{A}_t &= \mathbb{A}_{t-1}[x_t]/(\mu_t(x_t)) \\
&\cong \mathbb{B}_{t-1}[x_t]/(\pi_{\mathbb{A}_{t-1}\to\mathbb{B}_{t-1}}(\mu_t(x_t))) \\
&\cong \hat{\mathbb{A}}_{t-1}[x_t]/(\pi_{\mathbb{A}_{t-1}\to\hat{\mathbb{A}}_{t-1}}(\mu_t(x_t))) \oplus \bigoplus_{j=1}^{t-1}\bigoplus_{k=2}^{l_j} \mathbb{A}_{t-1}^{j,k}[x_t]/(\pi_{\mathbb{A}_{t-1}\to\mathbb{A}_{t-1}^{j,k}}(\mu_t(x_t))),
\end{aligned}$$

whose right-hand side contains $(a_{t-1,1},(a_{j,k})_{j\leqslant t-1,2\leqslant k\leqslant l_j})$. We finally recover the image of this element in $\mathbb{A}_t$ using d_t recursive calls of the Chinese remaindering procedure. Altogether, this shows that there exists a universal constant c_3 with

$$\mathsf{R}_{\mathbb{K}}(d_1,\ldots,d_t) \leqslant c_3\,\mathsf{M}_{\mathbb{K}}(d_1,\ldots,d_{t-1};d_t)\log d_t + \mathsf{R}_{\mathbb{K}}(d_1,\ldots,d_{t-1})\,d_t. \quad (5.6)$$

By using $C>1$, inequality (5.5) and Proposition 2.6 give

$$\begin{aligned}
\mathsf{Q}_{\mathbb{K}}(d_1,\ldots,d_t) &= O\left(\sum_{i=1}^{t} C^{i-1}\,\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,d_{i+1}\cdots d_t\log d_t\right) \\
&= O(C^t\,\mathsf{M}_{\mathbb{K}}(d)\log\bar{d}).
\end{aligned}$$

Similarly, the relation (5.6) yields

$$\mathsf{R}_{\mathbb{K}}(d_1,\ldots,d_t) = O(C^t \mathsf{M}_{\mathbb{K}}(d)\log \bar{d}).$$

Finally, using our assumption that $d_i \geqslant 2$ for $i=1,\ldots,t$, the relation (5.4) implies that

$$\mathsf{F}_{\mathbb{K}}(d_1,\ldots,d_t) = O(C^t \mathsf{M}_{\mathbb{K}}(d)\log \bar{d}). \qquad \square$$

5.5. Panoramic evaluation

We are now ready the state the main complexity result of this section, in terms of the detailed cost functions from section 2.1.

THEOREM 5.5. *Let $(\mathbb{A}_i)_{i\leqslant t}$ be an explicitly separable tower of degree $d := \dim_{\mathbb{K}} \mathbb{A}_t$, let $\bar{d} := \max(d_1,\ldots,d_t)$, and let T be a computation tree over $\mathbb{K}$-algebras of detailed cost $(\tau_{\text{in}}, \tau_{\text{out}}, \tau_{\text{add}}, \tau_{\text{mul}}, \tau_{\text{div}})$. Then the panoramic evaluation of T over $\mathbb{A}$ using Algorithm 3.2, along with the computation of auxiliary data, requires*

$$O((\tau_{\text{add}}\, d + (\tau_{\text{out}} + \tau_{\text{mul}})\, C^t \mathsf{M}_{\mathbb{K}}(d) + (\tau_{\text{in}} + \tau_{\text{div}})\, C^t \mathsf{M}_{\mathbb{K}}(d)\log \bar{d})\log d)$$

operations in $\mathbb{K}$. Let $\mathbb{A} \cong \mathbb{D}_1 \oplus \cdots \oplus \mathbb{D}_\ell$ denote the panoramic splitting in the output. By means of the auxiliary data, conversions in both directions of this isomorphism can be done in time

$$O(C^t \mathsf{M}_{\mathbb{K}}(d)\log \bar{d}\log d).$$

Proof. The proof is very similar to the one of Theorem 4.2. It is straightforward whenever $\tau_{\text{div}} = 0$, so we may freely assume $\tau_{\text{div}} > 0$ from now. Using redundant representations, the directed evaluation of T over $\mathbb{A}_t$ requires the following operations in $\mathbb{K}$:

- $O(\tau_{\text{add}}\, d)$ operations for the nodes of type in $\mathbb{K}_0 \cup \mathbb{K}_1 \cup \{+,-\}$;
- $O(\tau_{\text{mul}}\, C^t \mathsf{M}_{\mathbb{K}}(d))$ operations for the nodes of type $\times$ by Proposition 2.6;
- $O(\tau_{\text{div}}\, C^t \mathsf{M}_{\mathbb{K}}(d)\log \bar{d})$ operations for the zero-tests and inversions by Lemma 5.1 and Proposition 5.2;
- $O(\tau_{\text{out}}\, C^t \mathsf{M}_{\mathbb{K}}(d))$ operations to reduce the output values, by Lemma 5.1.

At the end, the directed splitting

$$\mathbb{A} \cong \mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell \tag{5.7}$$

satisfies

$$\sum_{i=1}^{\ell} \dim_{\mathbb{K}} \mathbb{H}_i \leqslant d \quad \text{and} \quad \dim_{\mathbb{K}} \mathbb{H}_i \leqslant d/2 \text{ for } i=1,\ldots,\ell. \tag{5.8}$$

The finalization of the residual branches $\mathbb{H}_1,\ldots,\mathbb{H}_\ell$ takes time $O(C^t \mathsf{M}_{\mathbb{K}}(d)\log \bar{d})$ by Lemma 5.4. In step 2 of Algorithm 3.2 we next project the τ_{in} input values of the computation tree to the residual algebras $\mathbb{H}_1,\ldots,\mathbb{H}_\ell$. Again by Lemma 5.4, this can be done in time $O(\tau_{\text{in}}\, C^t \mathsf{M}_{\mathbb{K}}(d)\log \bar{d})$.

Let $\mathsf{C}(d)$ denote the cost function of one panoramic evaluation of T over a $\mathbb{K}$-algebra of dimension $\leqslant d$. The above analysis shows the existence of a universal constant c such that

$$\mathsf{C}(d) \;\leqslant\; c(\tau_{\text{add}}\, d + (\tau_{\text{out}} + \tau_{\text{mul}})\, C^t \mathsf{M}_{\mathbb{K}}(d) + (\tau_{\text{in}} + \tau_{\text{div}})\, C^t \mathsf{M}_{\mathbb{K}}(d)\log \bar{d}) + \sum_{i=1}^{\ell} \mathsf{C}(\dim_{\mathbb{K}} \mathbb{H}_i).$$

Unrolling this inequality at most $\left\lceil \frac{\log d}{\log 2} \right\rceil$ times, thanks to (5.8), the claimed complexity bound follows.

Let $\mathsf{D}(d)$ denote the cost function of the conversions between $\mathbb{A}$ and a panoramic splitting. Using Lemma 5.4, we conclude that

$$\begin{aligned}\mathsf{D}(d) &= O(C^t \mathsf{M}_{\mathbb{K}}(d) \log \bar{d}) + \sum_{i=1}^{\ell} \mathsf{D}(\dim_{\mathbb{K}} \mathbb{H}_i) \\ &= O(C^t \mathsf{M}_{\mathbb{K}}(d) \log \bar{d} \log d),\end{aligned}$$

again thanks to (5.8). □

6. PRIMITIVE TOWER REPRESENTATIONS

Before we turn to the more technical topic of speeding up computations in algebraic towers, we will have a short *intermezzo* to study primitive tower representations. The results in this section are essentially from [29, sections 3 and 4.1], although we slightly generalized some of them.

6.1. Evaluating polynomials at points in algebras

Let $\mathbb{A}$ be an effective $\mathbb{K}$-algebra and let $\mathbb{B}$ be an effective $\mathbb{A}$-algebra with an explicit basis $b_1, \ldots, b_r$. We start by recalling the main result from [29, section 3] about the evaluation of multivariate polynomials in $\mathbb{A}[x_1, \ldots, x_t]$ at points in $\mathbb{B}^t$. This can in particular be used in order to compute so-called multivariate modular compositions. We recall that $\mathsf{m}_{\mathbb{A}}$ (resp. $\mathsf{m}_{\mathbb{B}}$) stands for the number of operations in $\mathbb{K}$ that are required in order to multiply two elements in $\mathbb{A}$ (resp. $\mathbb{B}$). We assume that $\mathsf{m}_{\mathbb{B}} \geqslant r\, \mathsf{m}_{\mathbb{A}}$ and that additions and subtractions in $\mathbb{A}$ and $\mathbb{B}$ are cheaper than multiplications.

PROPOSITION 6.1. *Let $\mathbb{B}$ be an effective $\mathbb{A}$-algebra of rank r, whose basis is explicitly given. Let $A \in \mathbb{A}[x_1, \ldots, x_t]$ with $\deg_{x_i} A < d_i$ for $i = 1, \ldots, t$, and let $a_1, \ldots, a_t \in \mathbb{B}$. If $d := d_1 \cdots d_t = O(r)$, then $A(a_1, \ldots, a_t)$ can be computed by a straight-line program in time*

$$O(\mathsf{m}_{\mathbb{B}} \sqrt{d} + \mathsf{m}_{\mathbb{A}}\, r\, d^{\omega-1}).$$

Proof. The case $t = 1$ corresponds to the classical baby-step giant-step algorithm over $\mathbb{B}$; see for instance [29, section 3.1]. The extension to $t \geqslant 2$ has been designed in [29, section 3.2]. In [29, Proposition 3.3], the assumption $d_i \geqslant 2$ for $i = 1, \ldots, t$ was used for convenience. But if $d_i = 1$ for some i, then x_i does not actually appear in A, so discarding x_i is free of charge in the straight-line program model. □

6.2. Primitive tower representations

The main difference with [29] is that we will consider primitive tower over arbitrary $\mathbb{K}$-algebras $\mathbb{A}$ instead of fields $\mathbb{K}$. In the same spirit as section 3.7, it will also be convenient to also integrate the idea of redundant representations in the definition.

From an abstract point of view, given two effective $\mathbb{K}$-algebras $\mathbb{A}$ and $\mathbb{B}$, we may use elements in $\mathbb{B}$ as a redundant representation of elements in $\mathbb{A}$ if we have an effective monomorphism $\Phi_0 \colon \mathbb{A} \hookrightarrow \mathbb{B}$ and an effective epimorphism $\Psi_0 \colon \mathbb{B} \twoheadrightarrow \mathbb{A}$ such that $\Psi_0 \circ \Phi_0$ is the identity map of $\mathbb{A}$. We say that $\mathbb{A}$ is an *effective retract* of $\mathbb{B}$. We will write $\mathsf{C}(\mathbb{A} \to \mathbb{B})$ (resp. $\mathsf{C}(\mathbb{A} \leftarrow \mathbb{B})$) for the cost of one evaluation of Φ_0 (resp. Ψ_0), and we denote $\mathsf{C}(\mathbb{A} \leftrightarrow \mathbb{B}) := \max(\mathsf{C}(\mathbb{A} \to \mathbb{B}), \mathsf{C}(\mathbb{A} \leftarrow \mathbb{B}))$.

From now on, assume that $\mathbb{A}$ is a product of separable field extensions of $\mathbb{K}$. We consider a tower of ring extensions of $\mathbb{A}$ of the form $\mathbb{A}_0 := \mathbb{A}$, $\mathbb{A}_i := \mathbb{A}_{i-1}[x_i]/(\mu_i(x_i))$ for $i = 1, \ldots, t$, where μ_i is monic and explicitly separable of degree d_i, which means that there exists a Bézout relation $1 = \theta_i \mu_i' + \xi_i \mu_i$ with $\theta_i \in \mathbb{A}_{i-1}[x_i]_{<d_i}$ and $\xi_i \in \mathbb{A}_{i-1}[x_i]_{<d_i-1}$.

DEFINITION 6.2. *A **primitive tower representation** of $(\mathbb{A}_i)_{i\leqslant t}$ consists of:*

- *A $\mathbb{K}$-algebra $\mathbb{B}$, a monomorphism $\Phi_0 \colon \mathbb{A} \hookrightarrow \mathbb{B}$, and an epimorphism $\Psi_0 \colon \mathbb{B} \twoheadrightarrow \mathbb{A}$ such that $\Psi_0 \circ \Phi_0$ is the identity map of $\mathbb{A}$.*
- *Linear forms $b_1(x_1) := x_1$, $b_i(x_1, \ldots, x_i) := x_i + \lambda_i b_{i-1}(x_1, \ldots, x_{i-1})$ with $\lambda_i \in \mathbb{K}$, for $i = 2, \ldots, t$.*
- *Monic polynomials $\nu_i \in \mathbb{B}[y_i]$, for $i = 1, \ldots, t$.*
- *$\phi_{i,j} \in \mathbb{B}[y_i]_{<d_1 \cdots d_i}$ for $i = 1, \ldots, t$ and $j = 1, \ldots, i$.*

Writing β_i for the class of y_i in $\mathbb{B}_i := \mathbb{B}[y_i]/(\nu_i(y_i))$, we assume that these data satisfy the following properties for $i = 1, \ldots, t$:

W_1. *The following map Φ_i is a monomorphism that extends Φ_0:*

$$\begin{aligned} \Phi_i \colon \quad \mathbb{A}_i &\hookrightarrow \mathbb{B}_i \\ \alpha_j &\mapsto \phi_{i,j}(\beta_i) \qquad\qquad (j = 1, \ldots, i). \end{aligned}$$

W_2. *The following map Ψ_i with $\Psi_i \circ \Phi_i = \mathrm{Id}_{\mathbb{A}_i}$ is an epimorphism that extends Ψ_0:*

$$\begin{aligned} \Psi_i \colon \quad \mathbb{B}_i &\twoheadrightarrow \mathbb{A}_i \\ \beta_i &\mapsto b_i(\alpha_1, \ldots, \alpha_i). \end{aligned}$$

W_3. $\Phi_i(b_i(\alpha_1, \ldots, \alpha_i)) = \beta_i$.

*The polynomials $\phi_{i,j}$ are called **parametrizations** of the α_j in terms of β_i.*

6.3. Complexity of conversions

In terms of the triangular set $(T_i)_{i\leqslant t}$ defined in the introduction (T_i is the canonical preimage of μ_i in $\mathbb{A}[x_1, \ldots, x_i]$), an element $a \in \mathbb{A}_i$ is represented by a polynomial $A(\alpha_1, \ldots, \alpha_i)$ with $A \in \mathbb{A}[x_1, \ldots, x_i]$ defined modulo $(T_1, \ldots, T_i)$. So the conversion of $a \in \mathbb{A}_t$ into an element of $\mathbb{B}_t$ boils down to evaluating $\Phi_0(A)(\phi_{t,1}(y_t), \ldots, \phi_{t,t}(y_t))$ modulo $\nu_t(y_t)$. The backward conversion consists in evaluating the image $\Psi_0(B)$ of a univariate polynomial $B \in \mathbb{B}[y_t]_{<d}$ at $b_t(\alpha_1, \ldots, \alpha_t)$. Both conversions are instances of multivariate modular composition problems that can be handled using Proposition 6.1. More precisely:

PROPOSITION 6.3. *Assume that we are given:*

- *$(\mathbb{A}_i)_{i\leqslant t}$ in terms of the defining polynomials $(\mu_i)_{i\leqslant t}$, and*
- *$(\mathbb{B}_i)_{i\leqslant t}$ in terms of $\lambda_2, \ldots, \lambda_t$, $(\nu_i)_{i\leqslant t}$ and $(\phi_{i,j})_{j\leqslant i\leqslant t}$,*

such that $(\mathbb{B}_i)_{i\leqslant t}$ is a primitive tower representation of $(\mathbb{A}_i)_{i\leqslant t}$. Then there exist straight-line programs for which:

- *One evaluation of Φ_t takes $O(\mathsf{m}_{\mathbb{B}} d^{\omega}) + d\,\mathsf{C}(\mathbb{A} \to \mathbb{B})$ operations in $\mathbb{K}$.*
- *The images $\rho_i := \Phi_{i-1}(\mu_i)$ for $i = 1, \ldots, t$ can be computed in time $O(\mathsf{m}_{\mathbb{B}} d^{\omega}) + 2d\,\mathsf{C}(\mathbb{A} \to \mathbb{B})$.*
- *Given $\rho_1, \ldots, \rho_t$, one evaluation of Ψ_t can be done in time $O(\mathsf{m}_{\mathbb{B}} d^{\omega}) + d\,\mathsf{C}(\mathbb{A} \leftarrow \mathbb{B})$.*

Proof. If $d_i = 1$ for some i, then x_i does not actually appear in polynomial representations of elements in $\mathbb{A}_t$. In the computation tree model, we may suppress such indices without any cost. Without loss of generality we may therefore assume that $d_i \geqslant 2$ for $i = 1, \ldots, t$.

The complexity bound for one evaluation of Φ_i with $i \leqslant t$ is a direct consequence of Proposition 6.1. Indeed, any element in $\mathbb{A}_i$ can be represented as $A(\alpha_1, \ldots, \alpha_i)$, where $A \in \mathbb{A}[x_1, \ldots, x_i]$ admits partial degrees $< d_j$ in x_j for $j = 1, \ldots, i$. Now we can compute

$$B \;:=\; \Phi_0(A)(\phi_{i,1}(y_i), \ldots, \phi_{i,i}(y_i)) \bmod v_i(y_i) \;\in\; \mathbb{B}[y_i]_{<d_1 \cdots d_i}$$

in time

$$\begin{aligned} & O\Big(\mathsf{m}_{\mathbb{B}}(d_1 \cdots d_i)^{\frac{3}{2}+\left(\omega-\frac{3}{2}\right)} + \mathsf{m}_{\mathbb{B}}(d_1 \cdots d_i)^{\omega}\Big) + d_1 \cdots d_i \mathsf{C}(\mathbb{A} \to \mathbb{B}) \\ =\; & O(\mathsf{m}_{\mathbb{B}}(d_1 \cdots d_i)^{\omega}) + d_1 \cdots d_i \mathsf{C}(\mathbb{A} \to \mathbb{B}), \end{aligned}$$

since $\omega > 3/2$. We finally notice that $\Phi_i(A(\alpha_1, \ldots, \alpha_i)) = B(\beta_i)$.

For $i = 1, \ldots, t$, the map Φ_{i-1} extends coefficientwise into a map $\mathbb{A}_{i-1}[x_i] \hookrightarrow \mathbb{B}_{i-1}[x_i]$. We may thus convert the minimal polynomial $\mu_i \in \mathbb{A}_{i-1}[x_i]$ of α_i over $\mathbb{A}_{i-1}$ into a polynomial $\rho_i \in \mathbb{B}_{i-1}[x_i]$ in time

$$O(\mathsf{m}_{\mathbb{B}}(d_1 \cdots d_{i-1})^{\omega} d_i) + d_1 \cdots d_i \mathsf{C}(\mathbb{A} \to \mathbb{B}).$$

The computation of $\rho_1, \ldots, \rho_t$ then takes time $O(\mathsf{m}_{\mathbb{B}} d^{\omega}) + 2d\,\mathsf{C}(\mathbb{A} \to \mathbb{B})$, since $d_i \geqslant 2$ for all i.

In order to evaluate Ψ_t at an element b in $\mathbb{B}_t$, we write $B \in \mathbb{B}[y_t]_{<d}$ for the preimage of b and observe that

$$\Psi_t(b) = \Psi_0(B)(b_t(\alpha_1, \ldots, \alpha_t)) = \Psi_0(B)(\alpha_t + \lambda_t b_{t-1}(\alpha_1, \ldots, \alpha_{t-1})).$$

We first evaluate B at $\chi_t + \lambda_t \beta_{t-1}$ in $\mathbb{B}_{t-1}[\chi_t] = \mathbb{B}[\beta_{t-1}][\chi_t]$, where χ_t is the class of x_t in $\mathbb{B}_{t-1}[x_t]/(\rho_t(x_t))$. This takes time $O(\mathsf{m}_{\mathbb{B}} d^{\omega})$ by Proposition 6.1 and yields a polynomial

$$\tilde{B}(x_t) := B(x_t + \lambda_t \beta_{t-1}) \bmod \rho_t(x_t).$$

Since $\Psi_{t-1} \circ \Phi_{t-1} = \mathrm{Id}_{\mathbb{A}_{t-1}}$, we next notice that

$$\Psi_{t-1}(\tilde{B}(x_t)) = \Psi_0(B)(x_t + \lambda_t \Psi_{t-1}(\beta_{t-1})) \bmod \mu_t(x_t),$$

so the preimage in $\mathbb{A}_{t-1}[x_t]$ of $\Psi_{t-1}(\tilde{B}(x_t))$ is

$$\Psi_0(B)(x_t + \lambda_t b_{t-1}(\alpha_1, \ldots, \alpha_{t-1})) \bmod \mu_t(x_t),$$

which represents $\Psi_t(b)$. Applying Ψ_{t-1} to the $\leqslant d_t$ coefficients of $\tilde{B}$ costs $d_t \mathsf{C}(\mathbb{A}_{t-1} \leftarrow \mathbb{B}_{t-1})$. Altogether, this proves the existence of a universal constant c such that

$$\mathsf{C}(\mathbb{A}_t \leftarrow \mathbb{B}_t) \leqslant c \mathsf{m}_{\mathbb{B}} d^{\omega} + d_t \mathsf{C}(\mathbb{A}_{t-1} \leftarrow \mathbb{B}_{t-1}).$$

We conclude that $\Psi_t(b)$ can be computed in time

$$\begin{aligned} & O(\mathsf{m}_{\mathbb{B}} d^{\omega} + d_t \mathsf{m}_{\mathbb{B}}(d/d_t)^{\omega} + \cdots + d_2 \cdots d_t \mathsf{m}_{\mathbb{B}}(d/(d_2 \cdots d_t))^{\omega}) + d\,\mathsf{C}(\mathbb{A} \leftarrow \mathbb{B}) \\ =\; & O(\mathsf{m}_{\mathbb{B}} d^{\omega}(1 + d_t^{1-\omega} + \cdots + (d_2 \cdots d_t)^{1-\omega})) + d\,\mathsf{C}(\mathbb{A} \leftarrow \mathbb{B}) \\ =\; & O(\mathsf{m}_{\mathbb{B}} d^{\omega}) + d\,\mathsf{C}(\mathbb{A} \leftarrow \mathbb{B}), \end{aligned}$$

using our assumptions that $\omega > 3/2$ and $d_i \geqslant 2$ for all i. □

6.4. Constructing primitive tower representations

If $\mathbb{A} = \mathbb{B} = \mathbb{K}$ is a field, then our definition of primitive tower representations of $(\mathbb{A}_i)_{i \leqslant t}$ coincides with the one from [29, Definition 2.2] whenever the Φ_i are isomorphisms. If $\mathbb{K}$ has sufficiently many elements, then primitive tower representations can for instance be computed using linear algebra techniques. For efficiency reasons, we will rely on the following result that will be used later with a parametric algebra in the role of $\mathbb{K}$.

PROPOSITION 6.4. [29, Corollary 4.4] *Assume that* $\mathbb{A}$ *is a field of cardinality* $\operatorname{card} \mathbb{A} > \binom{d}{2}$. *Then a primitive tower representation of* $(\mathbb{A}_i)_{i\leqslant t}$ *can be computed using* $O(d^2)$ *inversions and zero-tests in* $\mathbb{A}$, *plus* $O(d\,\mathsf{M}_{\mathbb{A}}(d^2)\log d)$ *ring operations in* $\mathbb{A}$.

Remark 6.5. Checking that the deterministic algorithm underlying Proposition 6.4 fits the computation tree model is straightforward and left up to the reader.

Remark 6.6. Proposition 6.4 admits a randomized variant [29, Corollary 4.4]: if $\operatorname{card} \mathbb{K} \geqslant 2\binom{d}{2}$, then a primitive tower representation of $(\mathbb{A}_i)_{i\leqslant t}$ can be computed by a randomized Las Vegas algorithm using an expected number of $O(d)$ inversions and zero-tests in $\mathbb{A}$ plus $O(\mathsf{M}_{\mathbb{K}}(d^2)\log d)$ arithmetic ring operations in $\mathbb{A}$.

It is more subtle to regard such randomized variants as computation trees. Although our model does not provide us with an instruction to compute random elements of $\mathbb{K}$, nothing prevents us from adding a "pool" of randomly chosen elements to the input. Since randomized Las Vegas algorithms may fail for certain random choices, they typically loop until suitable random numbers are drawn. As explained in Remark 2.4, we may unroll such loops to make things fit in our computation tree model, *provided* that we can provide an absolute bound on the number of times that we have to run the loop.

In our case, we claim that such an absolute bound indeed exists. Following [29, Proposition 4.3], the construction is incremental in t: assuming that $\lambda_2, \ldots, \lambda_{t-1}$ already found, one has to pick up a suitable value of λ_t in a subset $\mathcal{S}$ of $\mathbb{K}$ of cardinality $2\binom{d}{2}$. The crucial observation is that a suitable value always exists in this finite set $\mathcal{S}$, which bounds the number of random values that have to be tried. With $\mathcal{S}$ ordered randomly, λ_t can actually be found with a bounded number of expected trials.

7. ACCELERATED TOWER ARITHMETIC

The suboptimality of the complexity bounds of Proposition 2.6 and of Theorem 5.5 appears when the height t of the tower gets as large as $\lfloor \log_2 d \rfloor$. In fact, if t indeed gets large, then many of the d_i are necessarily small. This section is devoted to an efficient solution to this issue.

Roughly speaking, as long as a product of the form $d_{i+1} \cdots d_j$ is reasonably small, it turns out to be favorable to change the given representation of $\mathbb{A}_j = \mathbb{A}_i[\alpha_{i+1}, \ldots, \alpha_j]$ into a more efficient primitive element representation $\mathbb{A}_j \cong \mathbb{A}_i[\beta_j]$. Repeating this operation as many times as necessary, the original tower $(\mathbb{A}_i)_{i\leqslant t}$ is replaced by an equivalent tower $(\mathbb{B}_j)_{j\leqslant s}$ of much smaller height and for which all defining polynomials have sufficiently large degree. Such *accelerated towers* were first introduced in [29, section 4.2]. In this section, we will further refine the concept and show how to use such towers in combination with directed evaluation.

Throughout this section, we carry on with the notations from section 5: the original tower $(\mathbb{A}_i)_{i\leqslant t}$ is explicitly defined by $\mu_1, \ldots, \mu_t$, and we set $d_i := \deg \mu_i$ for $i = 1, \ldots, t$. As before, we will assume that $t \geqslant 2$ and that $d_i \geqslant 2$ for $i = 1, \ldots, t$. In particular, $d = d_1 \cdots d_t \geqslant 2$.

7.1. Accelerated towers

In this subsection, we extend the notion of accelerated towers from [29, Definition 4.6] and study accelerated tower arithmetic. Let $(\mathbb{A}_i)_{i\leqslant t}$ be an explicitly separable tower over $\mathbb{K}$, and let $\delta > 1$. By [29, Lemma 4.7], there exists a sequence of integers $0 = i_0 < i_1 < \cdots < i_s = t$ of length

$$s \leqslant 3\frac{\log d}{\log \delta} + 1, \tag{7.1}$$

such that, for $j=1,\dots,s$, we have

$$i_j > i_{j-1}+1 \implies e_j < \delta, \tag{7.2}$$

where $e_j := d_{i_{j-1}+1}\cdots d_{i_j}$. In this subsection, we assume that the sequence $i_0<\cdots<i_s$ has been fixed once and for all.

DEFINITION 7.1. *Let $r\in\{1,\dots,s\}$ and consider a factor tower $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$ of $(\mathbb{A}_i)_{i\leqslant i_r}$. Then the subsequence $(i_j)_{j\leqslant r}$ induces a family of towers $(\hat{\mathbb{A}}_{i_j})_{i_{j-1}+1\leqslant k\leqslant i_j}$ over $\hat{\mathbb{A}}_{i_{j-1}}$ for $j=1,\dots,r$. An **accelerated tower** for $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$, and with respect to the sub-sequence $(i_j)_{j\leqslant r}$, consists of a family of primitive tower representations of these towers $(\hat{\mathbb{A}}_{i_j})_{i_{j-1}+1\leqslant k\leqslant i_j}$ for $j=1,\dots,r$. More precisely, these representations comprise the following data:*

- *A tower $(\mathbb{B}_j)_{j\leqslant r}$ over $\mathbb{K}$ with defining polynomials $\nu_{i_1},\dots,\nu_{i_r}$, where $\mathbb{B}_j := \mathbb{B}_{j-1}[y_{i_j}]/(\nu_{i_j}(y_{i_j}))$ and $\nu_{i_j}\in\mathbb{B}_{j-1}[y_{i_j}]$, and such that $\deg\nu_{i_j}\leqslant e_j$. The class of y_{i_j} in $\mathbb{B}_j$ is written β_j.*
- *For $j=1,\dots,r$:*
  - *Linear forms $b_{i_{j-1}+1}=x_{i_{j-1}+1}$, $b_k=x_k+\lambda_k b_{k-1}$ over $\mathbb{K}$ for $k=i_{j-1}+2,\dots,i_j$,*
  - *$\nu_k\in\mathbb{B}_{j-1}[y_k]$ for $k=i_{j-1}+1,\dots,i_j-1$,*
  - *$\phi_{k,l}\in\mathbb{B}_{j-1}[y_k]_{<d_{i_{j-1}+1}\cdots d_k}$ for $k=i_{j-1}+1,\dots,i_j$ and $l=i_{j-1}+1,\dots,k$.*

Let Φ_0 represent the identity map of $\mathbb{K}$. For $j=1,\dots,r$ and $k=i_{j-1}+1,\dots,i_j$, these data induce the following monomorphism that extends $\Phi_{i_{j-1}}$:

$$\begin{aligned}\Phi_k\colon\quad \hat{\mathbb{A}}_k &\hookrightarrow \mathbb{B}_{j-1}[y_k]/(\nu_k(y_k))\\ \hat{\alpha}_l &\mapsto \phi_{k,l}(\beta_k) \text{ for } l=i_{j-1}+1,\dots,k.\end{aligned}$$

Let Ψ_0 represent the identity map of $\mathbb{K}$. For $j=1,\dots,r$ and $k=i_{j-1}+1,\dots,i_j$, we also have the following epimorphism that extends $\Psi_{i_{j-1}}$:

$$\begin{aligned}\Psi_k\colon\quad \mathbb{B}_{j-1}[y_k]/(\nu_k(y_k)) &\twoheadrightarrow \hat{\mathbb{A}}_k\\ \beta_k &\mapsto b_k(\hat{\alpha}_{i_{j-1}+1},\dots,\hat{\alpha}_k),\end{aligned}$$

and $\Psi_k\circ\Phi_k=\mathrm{Id}_{\hat{\mathbb{A}}_k}$. For $j=1,\dots,r$ and $k=i_{j-1}+1,\dots,i_j$, we finally have

$$\Phi_k(b_k(\alpha_{i_{j-1}-1},\dots,\alpha_k)) = \beta_k.$$

In particular, for $j=1,\dots,r$, we have the following monomorphism that extends $\Phi_{i_{j-1}}$:

$$\begin{aligned}\Phi_{i_j}\colon\quad \hat{\mathbb{A}}_{i_j} &\hookrightarrow \mathbb{B}_j=\mathbb{B}_{j-1}[y_{i_j}]/(\nu_{i_j}(y_{i_j}))\\ \hat{\alpha}_l &\mapsto \phi_{i_j,l}(y_{i_j}) && (l=i_{j-1}+1,\dots,i_j)\\ b_{i_j}(\hat{\alpha}_{i_{j-1}+1},\dots,\hat{\alpha}_{i_j}) &\mapsto \beta_{i_j}.\end{aligned}$$

7.1.1. Accelerated arithmetic

When $r=s$, taking

$$\delta := d^{\epsilon(d)}, \qquad \epsilon(d) := \sqrt{\frac{3\log C}{2\log d}} \tag{7.3}$$

yields

$$s = O\left(\sqrt{\frac{\log d}{\log C}}\right), \qquad C^s = \exp\left(O\left(\sqrt{\log d}\right)\right) = d^{O(1/\sqrt{\log d})}. \tag{7.4}$$

whence products in $\mathbb{B}_s$ can be computed using $d^{O(1/\sqrt{\log d})} = O(d^{1+o(1)})$ operations in $\mathbb{K}$, by Proposition 2.6. The worst case complexity of products in $(\mathbb{B}_j)_{j\leqslant s}$ is therefore asymptotically better than in $(\hat{\mathbb{A}}_i)_{i\leqslant t}$, which explains the terminology of "accelerated towers". Let us now detail how to make elements in $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$ benefit from this acceleration.

LEMMA 7.2. *Let $(\mathbb{B}_j)_{j\leqslant r}$ be an accelerated tower for $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$ as in Definition 7.1. Given $j\leqslant r$ and $i_{j-1}<i\leqslant i_j$, conversions between $\mathbb{B}_{j-1}[\beta_i]=\mathbb{B}_{j-1}[y_i]/(\nu_i(y_i))$ and $\hat{\mathbb{A}}_i$ via Φ_i and Ψ_i can be done by straight-line programs in time $O(C^j \mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1})$.*

Proof. The proof is adapted from [29, Lemma 4.8]. Let us first assume that the polynomials $\rho_k := \Phi_{k-1}(\mu_k)$ have been precomputed for $k=1,\ldots,i$. By Proposition 2.6,

$$\mathsf{m}_{\mathbb{B}_{j-1}} \;=\; O(C^{j-1}\mathsf{M}_{\mathbb{K}}(e_1\cdots e_{j-1})),$$

and by Proposition 6.3, we have

$$\begin{aligned}\mathsf{C}(\hat{\mathbb{A}}_i \leftrightarrow \mathbb{B}_{j-1}[\beta_i]) &= O(\mathsf{m}_{\mathbb{B}_{j-1}}(d_{i_{j-1}+1}\cdots d_i)^{\omega}) + \mathsf{C}(\mathbb{A}_{i_{j-1}} \leftrightarrow \mathbb{B}_{j-1})\, d_{i_{j-1}+1}\cdots d_i\\ &= O(C^{j-1}\mathsf{M}_{\mathbb{K}}(e_1\cdots e_{j-1})\,(d_{i_{j-1}+1}\cdots d_i)^{\omega}) + \mathsf{C}(\mathbb{A}_{i_{j-1}} \leftrightarrow \mathbb{B}_{j-1})\, d_{i_{j-1}+1}\cdots d_i.\end{aligned}$$

If $i_j = i_{j-1}+1$ then such conversions simply cost $\mathsf{C}(\mathbb{A}_{i_{j-1}} \leftrightarrow \mathbb{B}_{i_{j-1}})\, d_{i_{j-1}+1}\cdots d_i$. Otherwise we have $e_j<\delta$ by (7.2). In both cases it follows that

$$\mathsf{C}(\hat{\mathbb{A}}_i \leftrightarrow \mathbb{B}_{j-1}[\beta_i]) \;\leqslant\; cC^{j-1}\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1} + \mathsf{C}(\mathbb{A}_{i_{j-1}} \leftrightarrow \mathbb{B}_{j-1})\, d_{i_{j-1}+1}\cdots d_i,$$

for some universal constant c. Unrolling the latter inequality, while taking into account that $\mathbb{A}_0=\mathbb{B}_0=\mathbb{K}$ and $C>1$, we deduce that

$$\begin{aligned}\mathsf{C}(\hat{\mathbb{A}}_i \leftrightarrow \mathbb{B}_{j-1}[\beta_i]) &\leqslant cC^{j-1}\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1}\\ &\quad + d_{i_{j-1}+1}\cdots d_i\,(cC^{j-2}\mathsf{M}_{\mathbb{K}}(e_1\cdots e_{j-1})\,\delta^{\omega-1} + e_{j-1}\,\mathsf{C}(\mathbb{A}_{i_{j-2}} \leftrightarrow \mathbb{B}_{j-2}))\\ &= c\,(C^{j-1}+C^{j-2})\,\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1} + \mathsf{C}(\mathbb{A}_{i_{j-2}} \leftrightarrow \mathbb{B}_{j-2})\, e_{j-1} d_{i_{j-1}+1}\cdots d_i\\ &\leqslant \cdots\\ &= O(C^{j-1}\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1}).\end{aligned}$$

Finally, given $k\in\{i_{j-1}+1,\ldots,i\}$, we notice that $\rho_k := \Phi_{k-1}(\mu_k)$ can be precomputed in time

$$O(\mathsf{m}_{\mathbb{B}_{j-1}}(d_{i_{j-1}+1}\cdots d_k)^{\omega}) + 2\,\mathsf{C}(\mathbb{A}_{i_{j-1}} \leftrightarrow \mathbb{B}_{j-1})\, d_{i_{j-1}+1}\cdots d_k = O(C^j\mathsf{M}_{\mathbb{K}}(d_1\cdots d_k)\,\delta^{\omega-1}),$$

by Proposition 6.3, again by distinguishing the cases when $i_j = i_{j-1}+1$ and $i_j > i_{j-1}+1$. Consequently, the total cost of these precomputations is

$$O\left(\sum_{l=1}^{j}\sum_{k=i_{l-1}+1}^{\min(i,i_l)} C^l\mathsf{M}_{\mathbb{K}}(d_1\cdots d_k)\,\delta^{\omega-1}\right) = O(C^j\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1}). \qquad \square$$

PROPOSITION 7.3. *Let $(\mathbb{B}_j)_{j\leqslant r}$ be an accelerated tower for $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$ as in Definition 7.1. Given $j\leqslant r$ and $i_{j-1}<i\leqslant i_j$, products in $\hat{\mathbb{A}}_i$ can be computed by a straight-line program in time*

$$\mathsf{m}_{\hat{\mathbb{A}}_i} = O(C^j\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1}).$$

Two polynomials of degree $\leqslant n$ over $\hat{\mathbb{A}}_i$ can be multiplied by a straight-line program in time

$$\mathsf{M}_{\hat{\mathbb{A}}_i}(n) = O(C^j(\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1}n + \mathsf{M}_{\mathbb{K}}(d_1\cdots d_i n))).$$

Proof. One conversion between $\hat{\mathbb{A}}_i$ and $\mathbb{B}_{j-1}[\beta_i]$ can be done in time

$$\mathsf{C}(\hat{\mathbb{A}}_i \leftrightarrow \mathbb{B}_{j-1}[\beta_i]) \;=\; O(C^{j-1}\mathsf{M}_{\mathbb{A}}(d_1\cdots d_i)\,\delta^{\omega-1})$$

by Lemma 7.2. On the other hand, Proposition 2.6 yields

$$\begin{aligned}\mathsf{m}_{\mathbb{B}_{i_{j-1}}[\beta_i]} &= O(C^j\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i))\\ \mathsf{M}_{\mathbb{B}_{i_{j-1}}[\beta_i]}(n) &= O(C^j\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i n)),\ \text{for all } n\geqslant 0.\end{aligned}$$

Combining the latter costs, we deduce that

$$\begin{aligned}\mathsf{M}_{\hat{\mathbb{A}}_i}(n) &= O(\mathsf{C}(\hat{\mathbb{A}}_i \leftrightarrow \mathbb{B}_{i_{j-1}}[\beta_i])\,n) + \mathsf{M}_{\mathbb{B}_{i_{j-1}}[\beta_i]}(n)\\ &= O(C^j(\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1}n + \mathsf{M}_{\mathbb{K}}(d_1\cdots d_i n))),\end{aligned}$$

and the claimed cost for $\mathsf{m}_{\hat{\mathbb{A}}_i}$ corresponds to setting $n=1$. □

7.1.2. Redundant representations

One motivation behind Definitions 6.2 and 7.1 with respect to the less general definitions from [29] is to allow for redundant representations. More precisely, we have:

LEMMA 7.4. *Let $(\mathbb{B}_j)_{j\leqslant r}$ be an accelerated tower for $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$ as in Definition 7.1. Then $(\mathbb{B}_j)_{j\leqslant r}$ is an accelerated tower for any factor tower of $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$.*

Proof. A tower factor $(\tilde{\mathbb{A}}_i)_{i\leqslant i_r}$ of $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$ is a tower factor of $(\mathbb{A}_i)_{i\leqslant i_r}$. Let $\iota_{\tilde{\mathbb{A}}_k\to\hat{\mathbb{A}}_k}\colon \tilde{\mathbb{A}}_k \hookrightarrow \hat{\mathbb{A}}_k$ denote the canonical monomorphism, then $\pi_{\hat{\mathbb{A}}_k\to\tilde{\mathbb{A}}_k}\circ\iota_{\tilde{\mathbb{A}}_k\to\hat{\mathbb{A}}_k}$ is the identity of $\tilde{\mathbb{A}}_k$. Setting $\tilde{\Phi}_k := \Phi_0$ we first verify by induction that

$$\begin{aligned}\tilde{\Phi}_k\colon\ \tilde{\mathbb{A}}_k &\hookrightarrow \mathbb{B}_{j-1}[y_k]/(\nu_k(y_k))\\ \tilde{\alpha}_l &\mapsto \phi_{k,l}(\beta_k) \qquad\qquad (l = i_{j-1}+1,\ldots,k)\end{aligned}$$

is a monomorphism that extends $\tilde{\Phi}_{i_{j-1}}$, for $k = i_{j-1}+1,\ldots,i_j$, since $\tilde{\Phi}_k = \Phi_k\circ\iota_{\tilde{\mathbb{A}}_k\to\hat{\mathbb{A}}_k}$.

Let $\tilde{\Psi}_0$ represent the identity map of $\mathbb{K}$. For $j=1,\ldots,r$, we define the epimorphism $\tilde{\Psi}_k$, for $k = i_{j-1}+1,\ldots,i_j$, that extends $\tilde{\Psi}_{i_{j-1}}$:

$$\begin{aligned}\tilde{\Psi}_k\colon\ \mathbb{B}_{j-1}[y_k]/(\nu_k(y_k)) &\twoheadrightarrow \tilde{\mathbb{A}}_k\\ \beta_k &\mapsto b_k(\tilde{\alpha}_{i_{j-1}+1},\ldots,\tilde{\alpha}_k).\end{aligned}$$

Since $\tilde{\Psi}_k = \pi_{\hat{\mathbb{A}}_k\to\tilde{\mathbb{A}}_k}\circ\Psi_k$, we deduce that $\tilde{\Psi}_k\circ\tilde{\Phi}_k$ is the identity map of $\tilde{\mathbb{A}}_k$. □

Notice that the tower $(\mathbb{B}_j)_{j\leqslant r}$ is only used for the acceleration of arithmetic operations; we do not require this tower to be separable. Writing $\hat{\mathbb{B}}_j$ for the image of $\hat{\mathbb{A}}_{i_j}$ in $\mathbb{B}_j$, our use of redundant representations also makes it unnecessary to determine the defining polynomials of $(\hat{\mathbb{B}}_j)_{j\leqslant r}$. Although the computation of such defining polynomials $\hat{\nu}_j$ along with Bézout relations for $\hat{\nu}_j$ and $\hat{\nu}_j'$ would be convenient for speeding up multiplications in $\hat{\mathbb{A}}_t$, it seems tricky to integrate these computations without deteriorating the overall complexity.

When reduced, non-redundant representations are explicitly required, we will resort to the following counterpart of Lemma 5.1:

LEMMA 7.5. *Let $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$ be a factor tower of $(\mathbb{A}_i)_{i\leqslant i_r}$, and let $(\mathbb{B}_j)_{j\leqslant r}$ be an accelerated tower for $(\hat{\mathbb{A}}_i)_{i\leqslant i_r}$ as in Definition 7.1. Given $j\leqslant r$ and $i_{j-1} < i \leqslant i_j$, the projection of an element $a\in\mathbb{A}_i$ onto $\hat{\mathbb{A}}_i$ can be computed by a straight-line program in time $O(C^j\mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1}\log\delta)$.*

Proof. The proof is the same as in Lemma 5.1, except that we appeal to accelerated products $\mathsf{M}_{\hat{\mathbb{A}}_i}(n) = O(C^j \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i n)\, \delta^{\omega-1})$ of Proposition 7.3. The cost of the projection thus becomes:

$$\begin{aligned}
\mathsf{C}(\mathbb{A}_i \to \hat{\mathbb{A}}_i) &= O\left(\sum_{k=1}^{i} \mathsf{M}_{\hat{\mathbb{A}}_{k-1}}(d_k)\, d_{k+1} \cdots d_i\right) \\
&= O\left(\sum_{l=1}^{j} \sum_{k=i_{l-1}+1}^{\min(i,i_l)} C^l \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i)\, \delta^{\omega-1}\right) \\
&= O\left(\sum_{l=1}^{j} C^l \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i)\, \delta^{\omega-1} (i_l - i_{l-1})\right).
\end{aligned}$$

Now (7.2) yields $i_l - i_{l-1} = O(\log \delta)$ for $l = 1, ..., j$. Since $C > 1$, the claimed bound follows. □

7.1.3. Directed zero-tests and inversions

PROPOSITION 7.6. *Let $(\hat{\mathbb{A}}_i)_{i \leqslant i_r}$ be a principal branch encountered in a directed evaluation over $(\mathbb{A}_i)_{i \leqslant i_{r'}}$ and let $(\mathbb{B}_j)_{j \leqslant r}$ be an accelerated tower of $(\hat{\mathbb{A}}_i)_{i \leqslant i_{r'}}$ as in Definition 7.1. Given $j \leqslant r$ and $i_{j-1} < i \leqslant i_j$, one directed zero-test or inversion can be done in time*

$$O(C^j \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i)\, \delta^{\omega-1} \log^2 \delta \log \bar{d}).$$

Proof. We adapt the proof of Proposition 5.2 so as to benefit from accelerated products. From Proposition 7.3 we have

$$\mathsf{M}_{\mathbb{K}}(d_1, \ldots, d_i; n) = O(C^j \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i n)\, \delta^{\omega-1})),$$

and from Lemma 7.5 we have

$$\mathsf{P}_{\mathbb{K}}(d_1, \ldots, d_i) = O(C^j \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i)\, \delta^{\omega-1} \log \delta).$$

Consequently, the relation (5.1) implies the existence of a universal constant c such that

$$\begin{aligned}
\mathsf{D}_{\mathbb{K}}(d_1, \ldots, d_i) &= O(\mathsf{M}_{\mathbb{K}}(d_1, \ldots, d_{i-1}; d_i) \log d_i + \mathsf{P}_{\mathbb{K}}(d_1, \ldots, d_{i-1})\, d_i) + \mathsf{D}_{\mathbb{K}}(d_1, \ldots, d_{i-1})\, d_i \\
&\leqslant c C^{j-1} \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i)\, \delta^{\omega-1} \log \delta \log d_i + \mathsf{D}_{\mathbb{K}}(d_1, \ldots, d_{i-1})\, d_i.
\end{aligned}$$

Unrolling the latter inequality yields

$$\begin{aligned}
\mathsf{D}_{\mathbb{K}}(d_1, \ldots, d_i) &= O\left(\sum_{l=1}^{j} \sum_{k=i_{l-1}+1}^{\min(i,i_l)} C^l \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i)\, \delta^{\omega-1} \log \delta \log d_k\right) \\
&= O\left(\sum_{l=1}^{j} C^l \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i)\, \delta^{\omega-1} \log^2 \delta \log \bar{d}\right) \\
&= O(C^j \mathsf{M}_{\mathbb{K}}(d_1 \cdots d_i)\, \delta^{\omega-1} \log^2 \delta \log \bar{d}),
\end{aligned}$$

while using the facts that $C > 1$ and $i_l - i_{l-1} = O(\log \delta)$ for $l = 1, \ldots, j$, by (7.2). □

7.2. Directed construction of an accelerated tower

Now that we have seen how to benefit from accelerated arithmetic, let us show how to construct accelerated towers. We use induction on the height, in combination with Proposition 6.4. As usual, all computations in the directed model are done using the redundant representation.

Algorithm 7.1

Input. An explicitly separable tower $(\mathbb{A}_i)_{i\leqslant t}$ and a sequence of integers $1\leqslant i_1<\cdots<i_s=t$.

Output. A directed splitting of $(\mathbb{A}_i)_{i\leqslant t}$ with principal branch $(\hat{\mathbb{A}}_i)_{i\leqslant t}$ and an accelerated tower $(\mathbb{B}_j)_{j\leqslant s}$ for $(\hat{\mathbb{A}}_i)_{i\leqslant t}$.

1. If $s=0$, then return $()$ for the directed splitting along with an empty accelerated tower.
2. Otherwise $s\geqslant 1$; recursively apply the algorithm to $(\mathbb{A}_i)_{i\leqslant i_{s-1}}$ and $1\leqslant i_1<\cdots<i_{s-1}$. Let $(\tilde{\mu}_i)_{i\leqslant i_{s-1}}$ represent the directed splitting of $(\mathbb{A}_i)_{i\leqslant i_{s-1}}$ obtained in return, let $(\tilde{\mathbb{A}}_i)_{i\leqslant i_{s-1}}$ be the principal branch, and let $(\mathbb{B}_j)_{j\leqslant s-1}$ be the accelerated tower for $(\tilde{\mathbb{A}}_i)_{i\leqslant i_{s-1}}$.
3. Let $P_{i_{s-1}+1},\dots,P_{i_s}$ be the canonical preimages of $\mu_{i_{s-1}+1},\dots,\mu_{i_s}$ in $\mathbb{A}_{i_{s-1}}[x_{i_{s-1}+1},\dots,x_{i_s}]$; compute $Q_{i_{s-1}+1}:=\pi_{\mathbb{A}_{i_{s-1}}\to\tilde{\mathbb{A}}_{i_{s-1}}}(P_{i_{s-1}+1}),\dots,Q_{i_s}:=\pi_{\mathbb{A}_{i_{s-1}}\to\tilde{\mathbb{A}}_{i_{s-1}}}(P_{i_s})$.
4. By directed evaluation over $\mathbb{A}_{i_{s-1}}$, starting with the directed splitting $(\tilde{\mu}_i)_{i\leqslant i_{s-1}}$, compute a primitive tower representation for
$$\tilde{\mathbb{A}}_{i_{s-1}}[x_{i_{s-1}+1}]/(Q_{i_{s-1}+1})\subseteq\cdots\subseteq\tilde{\mathbb{A}}_{i_{s-1}}[x_{i_{s-1}+1},\dots,x_{i_s}]/(Q_{i_{s-1}+1},\dots,Q_{i_s}),$$
seen as a tower over $\tilde{\mathbb{A}}_{i_{s-1}}$. Let $(\vec{\mu}_i)_{i\leqslant i_{s-1}}$ be the directed splitting obtained in return and let $(\hat{\mathbb{A}}_i)_{i\leqslant i_{s-1}}$ be the corresponding principal branch.
5. The latter primitive tower representation consists of
   - Linear forms $b_{i_{s-1}+1}=x_{i_{s-1}+1}$, $b_k=x_k+\lambda_k b_{k-1}$ over $\mathbb{K}$ for $k=i_{s-1}+2,\dots,i_s$;
   - $\nu_k^{\mathbb{A}}\in\hat{\mathbb{A}}_{i_{s-1}}[y_k]$ for $k=i_{j-1}+1,\dots,i_j$;
   - $\phi_{k,l}^{\mathbb{A}}\in\hat{\mathbb{A}}_{i_{s-1}}[y_k]_{<d_{i_{j-1}+1}\cdots d_k}$ for $k=i_{j-1}+1,\dots,i_j$ and $l=i_{j-1}+1,\dots,k$.

   With the notation of Definition 7.1, we set $\nu_k:=\Phi_{i_{s-1}}(\nu_k^{\mathbb{A}})$ for $k=i_{j-1}+1,\dots,i_j$ and $\phi_{k,l}:=\Phi_{i_{s-1}}(\phi_{k,l}^{\mathbb{A}})$ for $k=i_{j-1}+1,\dots,i_j$ and $l=i_{j-1}+1,\dots,k$.
6. Compute $\pi_{\tilde{\mathbb{A}}_{i_{s-1}}\to\hat{\mathbb{A}}_{i_{s-1}}}(Q_{i_{s-1}+1}),\dots,\pi_{\tilde{\mathbb{A}}_{i_s}\to\hat{\mathbb{A}}_{i_s}}(Q_{i_s})$ and define, for $k=i_{s-1}+1,\dots,i_s$:
   - $\hat{\mu}_k$ as $\pi_{\tilde{\mathbb{A}}_{i_{s-1}}\to\hat{\mathbb{A}}_{i_{s-1}}}(Q_{i_{s-1}+1})$ regarded as in $\hat{\mathbb{A}}_{k-1}[x_k]$;
   - $\hat{\mathbb{A}}_k:=\hat{\mathbb{A}}_{k-1}[y_k]/(\hat{\mu}_k(x_k))$.

   Complete the tower $(\hat{\mathbb{A}}_i)_{i\leqslant t}$ with the projections of the Bézout relations of μ_k' and μ_k over $\hat{\mathbb{A}}_{i_{s-1}}$; return this tower, $(\vec{\mu}_i)_{i\leqslant i_{s-1}},(\hat{\mu}_{i_{s-1}+1}),\dots,(\hat{\mu}_{i_s})$, and $(\mathbb{B}_j)_{j\leqslant s}$.

PROPOSITION 7.7. *If* $\operatorname{card}\mathbb{K}>\binom{d}{2}$, *then Algorithm 7.1 is correct and runs in time*
$$O(C^s\mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega+1}\log(\delta\bar{d})),$$
for a sequence of integers $(i_j)_{j\leqslant s}$ *that satisfies* (7.2).

Proof. When entering step 5, $(\mathbb{B}_j)_{j\leqslant s-1}$ is an accelerated tower for $(\hat{\mathbb{A}}_i)_{i\leqslant i_{s-1}}$; so we have the monomorphism $\Phi_{i_{s-1}}\colon\hat{\mathbb{A}}_{i_{s-1}}\hookrightarrow\mathbb{B}_{s-1}$ and the epimorphism $\Psi_{i_{s-1}}\colon\mathbb{B}_{s-1}\twoheadrightarrow\hat{\mathbb{A}}_{i_{s-1}}$ such that $\Psi_{i_{s-1}}\circ\Phi_{i_{s-1}}$ is the identity map of $\hat{\mathbb{A}}_{i_{s-1}}$. We also have the following isomorphisms for $k=i_{s-1}+1,\dots,i_s$:
$$\begin{aligned}\hat{\mathbb{A}}_k:=\hat{\mathbb{A}}_{i_{s-1}}[\hat{\alpha}_{i_{s-1}+1},\dots,\hat{\alpha}_k]&\cong\hat{\mathbb{A}}_{i_{s-1}}[y_k]/(\nu_k^{\mathbb{A}}(y_k))\\ \hat{\alpha}_l&\mapsto\phi_{k,l}^{\mathbb{A}}(y_k)\bmod\nu_k^{\mathbb{A}}(y_k)\qquad(l=i_{s-1}+1,\dots,k).\end{aligned}$$
So the following morphisms are monomorphisms that extend $\Phi_{i_{s-1}}\colon\hat{\mathbb{A}}_{i_{s-1}}\hookrightarrow\mathbb{B}_{s-1}$, for $k=i_{s-1}+1,\dots,i_s$:
$$\begin{aligned}\Phi_k\colon\hat{\mathbb{A}}_k&\hookrightarrow\mathbb{B}_{s-1}[y_k]/(\nu_k(y_k))=:\mathbb{B}_{s-1}[\beta_k]\\ \hat{\alpha}_l&\mapsto\phi_{k,l}(\beta_k)\qquad(l=i_{s-1}+1,\dots,k)\\ b_k(\hat{\alpha}_{i_{s-1}+1},\dots,\hat{\alpha}_k)&\mapsto\beta_k.\end{aligned}$$

Let

$$\begin{aligned}\Psi_k\colon\quad \mathbb{B}_{s-1}[\beta_k] &\twoheadrightarrow \hat{\mathbb{A}}_k\\ \beta_k &\mapsto b_k(\hat{\alpha}_{i_{s-1}+1},\ldots,\hat{\alpha}_k).\end{aligned}$$

We observe that Ψ_k extends $\Psi_{i_{s-1}}$ and that

$$\begin{aligned}(\Psi_k\circ\Phi_k)(\hat{\alpha}_l) &= \Psi_k(\phi_{k,l}(\beta_k))\\ &= \Psi_{i_{s-1}}(\phi_{k,l})(b_k(\hat{\alpha}_{i_{s-1}+1},\ldots,\hat{\alpha}_k))\\ &= \Psi_{i_{s-1}}(\Phi_{i_{s-1}}(\phi_{k,l}^{\mathbb{A}}))(b_k(\hat{\alpha}_{i_{s-1}+1},\ldots,\hat{\alpha}_k))\\ &= \phi_{k,l}^{\mathbb{A}}(b_k(\hat{\alpha}_{i_{s-1}+1},\ldots,\hat{\alpha}_k)) = \hat{\alpha}_l.\end{aligned}$$

This completes the correctness proof.

Step 1 takes constant time. In step 3, the projections require

$$O\left(\sum_{k=i_{s-1}+1}^{i_s} C^{s-1}\,\mathsf{M}_{\mathbb{K}}(d_1\cdots d_{i_{s-1}})\,d_{i_{s-1}+1}\cdots d_k\,\delta^{\omega-1}\log\delta\right) = O(C^s\,\mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1}\log\delta)$$

operations in $\mathbb{K}$, by Lemma 7.5 and the assumption that $d_i\geqslant 2$ for $i=1,\ldots,t$. Notice that the same bound holds for step 6.

Step 4 is free of charge whenever $i_s=i_{s-1}+1$. Otherwise $e_s\leqslant\delta$ holds by (7.2). During the directed evaluation of step 4, one product in the principal branch takes $O(C^{s-1}\,\mathsf{M}_{\mathbb{K}}(e_1\cdots e_{s-1})\,\delta^{\omega-1})$ operations in $\mathbb{K}$ by Proposition 7.3, and the cost of one directed inversion is $O(C^{s-1}\,\mathsf{M}_{\mathbb{K}}(e_1\cdots e_{s-1})\,\delta^{\omega-1}\log^2\delta\log\bar{d})$ by Proposition 7.6. Consequently, step 4 runs in time

$$\begin{aligned}&O(C^{s-1}\,\mathsf{M}_{\mathbb{K}}(e_1\cdots e_{s-1})\,e_s^2\,\delta^{\omega-1}\log^2\delta\log\bar{d} + C^{s-1}\,\mathsf{M}_{\mathbb{K}}(e_1\cdots e_{s-1}e_s^2)\,e_s\,\delta^{\omega-1}\log\delta)\\ =\;& O(C^{s-1}\,\mathsf{M}_{\mathbb{K}}(e_1\cdots e_s)\,e_s\,\delta^{\omega-1}\log^2\delta\log\bar{d} + C^{s-1}\,\mathsf{M}_{\mathbb{K}}(e_1\cdots e_s)\,e_s^2\,\delta^{\omega-1}\log\delta)\\ =\;& O(C^{s-1}\,\mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega}\log^2\delta\log\bar{d} + C^{s-1}\,\mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega+1}\log\delta)\\ =\;& O(C^s\,\mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega+1}\log(\delta\bar{d})),\end{aligned}$$

by Proposition 6.4 and (2.1).

In step 5 we perform

$$O(d_{i_{s-1}+1} + 2\,d_{i_{s-1}+1}d_{i_{s-1}+2} + \cdots + (i_s-i_{s-1})\,d_{i_{s-1}+1}\cdots d_{i_s}) = O(e_s\log\delta)$$

conversions from $\hat{\mathbb{A}}_{i_{s-1}}$ to $\mathbb{B}_{s-1}$, which amount to

$$O(C^{s-1}\,\mathsf{M}_{\mathbb{K}}(e_1\cdots e_{s-1})\,e_s\,\delta^{\omega-1}\log\delta) = O(C^s\,\mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1}\log\delta)$$

operations in $\mathbb{K}$ by Lemma 7.2. □

Remark 7.8. If $\operatorname{card}\mathbb{K}\geqslant 2\binom{d}{2}$, then the randomized Las Vegas version of Proposition 6.4 (see Remark 6.6) may be used in Algorithm 7.1, so that step 4 has an expected cost

$$O(C^s\,\mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega}\log(\delta\bar{d})),$$

which still dominates the cost of Algorithm 7.1 in this model.

7.3. Panoramic evaluation with accelerated towers

In order to make the directed evaluation benefit from accelerated arithmetic, we first need to compute an accelerated tower. Then, the complexity of accelerated divisions is simply adjusted according to Proposition 7.6. At the end of a computation, we use the following lemma in order to recover the extended tower factorization of $(\mathbb{A}_i)_{i\leqslant t}$, as in section 5.4.

LEMMA 7.9. *Given a directed splitting $(\overrightarrow{\mu_i})_{i\leqslant t}$ as above, and an accelerated tower for its principal branch, the explicitly separable factor towers $(\mathbb{A}_i^{j,k})_{i\leqslant t}$ for $j=1,\ldots,t$ and $k=1,\ldots,l_j$ can be computed in time $O(C^s \mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1}\log\delta\log\bar{d})$ by a straight-line program, where $\bar{d}:=\max(d_1,\ldots,d_t)$. In addition, both directions of the isomorphism*

$$\mathbb{A}_t \;\cong\; \hat{\mathbb{A}}_t \;\oplus\; \bigoplus_{j=1}^{t}\bigoplus_{k=2}^{l_j} \mathbb{A}_t^{j,k}$$

can be computed by straight-line programs in time $O(C^s \mathsf{M}_{\mathbb{K}}(d)\delta^{\omega-1}\log\delta\log\bar{d})$.

Proof. The algorithm is the same as in the proof of Lemma 5.4. We only revisit the cost analysis when using accelerated arithmetic. Given $i_{j-1}<i\leqslant i_j$, Proposition 7.3 yields

$$\mathsf{M}_{\mathbb{K}}(d_1,\ldots,d_i;n) = O(C^j \mathsf{M}_{\mathbb{K}}(d_1\cdots d_i\, n)\,\delta^{\omega-1}),$$

whence

$$\begin{aligned}\mathsf{Q}_{\mathbb{K}}(d_1,\ldots,d_t) \;&=\; O\left(\sum_{j=1}^{s}\sum_{i=i_{s-1}+1}^{i_s} C^j \mathsf{M}_{\mathbb{K}}(d_1\cdots d_i)\,\delta^{\omega-1} d_{i+1}\cdots d_t \log d_i\right)\\ &=\; O(C^s \mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1}\log\delta\log\bar{d}).\end{aligned}$$

In a similar way, we obtain

$$\mathsf{R}_{\mathbb{K}}(d_1,\ldots,d_t) \;=\; O(C^s \mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1}\log\delta\log\bar{d}).$$

We deduce that

$$\mathsf{F}_{\mathbb{K}}(d_1,\ldots,d_t) \;=\; O(C^s \mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1}\log\delta\log\bar{d}). \qquad \square$$

THEOREM 7.10. *Let $(\mathbb{A}_i)_{i\leqslant t}$ be an explicitly separable tower of degree $d:=\dim_{\mathbb{K}}\mathbb{A}_t$, let $\bar{d}:=\max(d_1,\ldots,d_t)$, and let T be a computation tree over $\mathbb{K}$-algebras of detailed cost $(\tau_{\text{in}},\tau_{\text{out}},\tau_{\text{add}},\tau_{\text{mul}},\tau_{\text{div}})$, as defined in section 2.1. Assume that* $\operatorname{card}\mathbb{K}>\binom{d}{2}$. *Then the panoramic evaluation of T over $\mathbb{A}_t$ using Algorithm 3.2, along with the computation of auxiliary data, requires*

$$O((\tau_{\text{add}}\,d+\sigma C^s \mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1})\log d) \tag{7.5}$$

operations in $\mathbb{K}$, where

$$\sigma=\tau_{\text{mul}}+\tau_{\text{div}}\log^2\delta\log\bar{d}+\tau_{\text{out}}\log\delta+\tau_{\text{in}}\log\delta\log\bar{d}+\delta^2\log(\delta\bar{d}).$$

Let $\mathbb{A}\cong\mathbb{D}_1\oplus\cdots\oplus\mathbb{D}_\ell$ denote the panoramic splitting in the output. Using the auxiliary data, conversions in both directions of this isomorphism can be done in time

$$O(C^s \mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1}\log\delta\log\bar{d}\log d). \tag{7.6}$$

Proof. The directed construction of the accelerated tower takes

$$O(C^s \mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega+1}\log(\delta\bar{d}))$$

by Proposition 7.7. Then, we project the input values onto the principal branch, written $(\hat{\mathbb{A}}_i)_{i\leqslant t}$, in time

$$O(\tau_{\text{in}} C^s \mathsf{M}_{\mathbb{K}}(d)\,\delta^{\omega-1}\log\delta),$$

by Lemma 7.5.

Let us summarize the number of operations in $\mathbb{K}$ needed for the directed evaluation of T over $\mathbb{A}_t$, when using redundant representations in $\hat{\mathbb{A}}_t$:

- $O(\tau_{\text{add}}\,d)$ operations for the nodes of type in $\mathbb{K}_0\cup\mathbb{K}_1\cup\{+,-\}$;

- $O(\tau_{\mathrm{mul}} C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega-1})$ for the nodes of type $\times$ by Proposition 7.3;
- $O(\tau_{\mathrm{div}} C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega-1} \log^2 \delta \log \bar{d})$ for the zero-tests and inversions by Lemma 7.5 and Proposition 7.6,
- $O(\tau_{\mathrm{out}} C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega-1} \log \delta)$ for the reduction of the output node by Lemma 7.5.

At the end, the directed splitting $\mathbb{A}_t \cong \mathbb{D} \oplus \mathbb{H}_1 \oplus \cdots \oplus \mathbb{H}_\ell$ satisfies

$$\sum_{i=1}^{\ell} \dim_{\mathbb{K}} \mathbb{H}_i \leqslant d, \qquad \dim_{\mathbb{K}} \mathbb{H}_i \leqslant d/2 \ \ (i=1,\ldots,\ell). \tag{7.7}$$

Then we finalize the construction of the residual branches $\mathbb{H}_1, \ldots, \mathbb{H}_\ell$ in time

$$O(C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega-1} \log \delta \log \bar{d}),$$

by Proposition 7.9. We next project the τ_{in} input values of the tree to $\mathbb{H}_1, \ldots, \mathbb{H}_\ell$, in time

$$O(\tau_{\mathrm{in}} C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega-1} \log \delta \log \bar{d}),$$

again by Proposition 7.9. We finally perform the recursive panoramic evaluations of T over $\mathbb{H}_1, \ldots, \mathbb{H}_\ell$. The conclusion follows as in the proof of Theorem 5.5. □

Remark 7.11. If $\mathrm{card}\, \mathbb{K} > 2\binom{d}{2}$, then the factor δ^2 in the definition of σ can be replaced by an expected δ in Theorem 7.10, by using the randomized variant from Remark 7.8 for the directed construction of the accelerated tower.

The following corollary simplifies the above complexity bound by tuning the parameters δ and s.

COROLLARY 7.12. *For a suitable choice of δ, the bound* (7.5) *simplifies into*

$$O(\tau_{\mathrm{add}}\, d \log d) + (\tau_{\mathrm{in}} + \tau_{\mathrm{out}} + \tau_{\mathrm{mul}} + \tau_{\mathrm{div}})\, d^{1+O(1/\sqrt{\log d})}$$

and the bound (7.6) *into $d^{1+O(1/\sqrt{\log d})}$.*

Proof. It suffices to take δ and s as in (7.3) and (7.4). □

7.4. Applications

The main result of [29] is an efficient algorithm to compute products in separable towers of field extensions. We are now in a position to generalize this result to arbitrary explicitly separable towers.

THEOREM 7.13. *Let $(\mathbb{A}_i)_{i \leqslant t}$ be an explicitly separable tower as in the introduction of this section. If* $\mathrm{card}\, \mathbb{K} > \binom{d}{2}$, *then products in $\mathbb{A}_t$ can be computed in time*

$$d^{1+O(1/\sqrt{\log d})}.$$

Proof. Let $a, b \in \mathbb{A}_t$. Applying Theorem 7.10 to the straight-line program that simply computes a product, it requires $O(C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega+1} \log(\delta \bar{d}) \log d)$ operations to compute a parametric splitting $\mathbb{A}_t \cong \mathbb{D}_1 \oplus \cdots \oplus \mathbb{D}_\ell$, auxiliary data, and the projections $\pi_{\mathbb{A} \to \mathbb{D}_i}(a\, b)$ for $i = 1, \ldots, \ell$. Using the auxiliary data, the theorem also allows us to recover $a\, b$ from these projections using $O(C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega-1} \log \delta \log \bar{d} \log d)$ further operations in $\mathbb{K}$. We conclude by taking δ and s as in (7.3) and (7.4). □

Another interesting application that has already been discussed in the univariate case at the end of section 4.3.3 is the computation of determinants.

THEOREM 7.14. *Let $(\mathbb{A}_i)_{i\leqslant t}$ be an explicitly separable tower as in the introduction of this section. If* $\operatorname{card}\mathbb{K} > \binom{d}{2}$*, then the determinant of an $n\times n$ matrix in $\mathbb{A}_t^{n\times n}$ can be computed in time*

$$n^\omega d^{1+O(1/\sqrt{\log d})}.$$

Proof. The proof is similar to the one of Theorem 7.13. □

As a final application, we revisit the matrix product. We need the following generalization of Proposition 2.6:

PROPOSITION 7.15. *With the notations of Proposition 2.6 and $C\geqslant 2$, assume that* $\operatorname{card}\mathbb{K}\geqslant 2^t d$*. Then the product of two $n\times n$ matrices in $\mathbb{A}_t^{n\times n}$ can be computed in time*

$$O(2^t n^\omega d + 2^t n^2 \mathsf{M}_{\mathbb{K}}(d)\log d + C^t n^2 \mathsf{M}_{\mathbb{K}}(d)).$$

Proof. We proceed along the same lines as Lebreton in [35, section 3.2]; see also [29, Proposition 2.4]. Consider two matrices $M, N\in\mathbb{A}_t^{n\times n}$, we first lift the matrices to multivariate polynomial matrices in $\mathbb{K}[x_1,\ldots,x_t]^{n\times n}$ of partial degrees $<d_i$ in x_i for $i=1,\ldots,t$. We next convert these polynomial matrices to univariate polynomial matrices in $\mathbb{K}[y]_{<2^{t-1}d}$ using Kronecker substitution. We multiply these matrices using the evaluation-interpolation technique. Since $\mathbb{K}$ admits at least $\operatorname{card}\mathbb{K}\geqslant 2^t d$ distinct evaluation points, this can be done in time

$$O(2^t n^\omega d + n^2 \mathsf{M}_{\mathbb{K}}(2^t d)\log(2^t d)).$$

We finally convert the result back to a polynomial matrix in $\mathbb{K}[x_1,\ldots,x_t]^{n\times n}$, and we reduce each of the entries with respect to $\mu_1,\ldots,\mu_t$ using the algorithm from [35, section 3.2]. This reduction step requires $O(n^2\mathsf{M}_{\mathbb{K}}(C^t d))$ operations in $\mathbb{K}$. Now our assumption that $d_i\geqslant 2$ for $i=1,\ldots,t$ implies $2^t\leqslant d$ and $C^t\leqslant d^2$. Using (2.1), we conclude that the total running time is bounded by

$$\begin{aligned} & O(2^t n^\omega d + n^2 \mathsf{M}_{\mathbb{K}}(2^t d)\log(2^t d) + n^2 \mathsf{M}_{\mathbb{K}}(C^t d)) \\ = \; & O(2^t n^\omega d + 2^t n^2 \mathsf{M}_{\mathbb{K}}(d)\log d + C^t n^2 \mathsf{M}_{\mathbb{K}}(d)). \end{aligned}$$ □

The next proposition is an example of a less straightforward use of Theorem 7.10, where we adapt the "acceleration factor" δ to a specific situation, and where we wish to decrease the overhead of the accelerated towers. Precisely, after a directed construction of an accelerated tower, we obtain a directed splitting $\mathbb{A}_t\cong\mathbb{D}\oplus\mathbb{H}_1\oplus\cdots\oplus\mathbb{H}_h$ that satisfies

$$\sum_{i=1}^{h}\dim_{\mathbb{K}}\mathbb{H}_i\leqslant d, \qquad \dim_{\mathbb{K}}\mathbb{H}_i\leqslant d/2 \quad (i=1,\ldots,h).$$

As a byproduct, we have an accelerated tower of $\mathbb{D}$ of degree $\leqslant d$ at our disposal. The recursive calls to panoramic evaluations over $\mathbb{H}_i$ for $i=1,\ldots,h$ then lead to accelerated towers for each of the factor towers of $(\mathbb{A}_i)_{i\leqslant t}$. Let $\mathsf{A}(d)$ denote a bound for the sum of the degrees of all these accelerated towers. Then we have

$$\mathsf{A}(d)\leqslant d+\mathsf{A}(\dim_{\mathbb{K}}\mathbb{H}_1)+\cdots+\mathsf{A}(\dim_{\mathbb{K}}\mathbb{H}_h).$$

Unrolling this inequality leads to $\mathsf{A}(d)=O(d\log d)$, which is slightly suboptimal.

We may modify the construction of accelerated towers, described in Algorithm 7.1, in order to ensure that the degree of the accelerated tower is never more than twice the degree of the principal branch before entering the evaluation of the given computation tree. This is done as follows. At the end of Algorithm 7.1, we compare the degree of the accelerated tower $(\mathbb{B}_j)_{j\leqslant s}$ to the degree of the principal branch $(\hat{\mathbb{A}}_i)_{i\leqslant t}$. If $\dim_{\mathbb{K}} \mathbb{B}_s \leqslant 2\dim_{\mathbb{K}} \hat{\mathbb{A}}_t$ then we are done. Otherwise, the principal branch is much smaller than $\mathbb{A}_t$ in the sense that $2\dim_{\mathbb{K}} \hat{\mathbb{A}}_t < \dim_{\mathbb{K}} \mathbb{B}_s \leqslant d$, and we use again Algorithm 7.1 in order to compute an accelerated tower for $(\hat{\mathbb{A}}_i)_{i\leqslant t}$. We carry on computing a new accelerated tower of the current principal branch in the directed model over $(\mathbb{A}_i)_{i\leqslant t}$ until the condition $\dim_{\mathbb{K}} \mathbb{B}_s \leqslant 2\dim_{\mathbb{K}} \hat{\mathbb{A}}_t$ is met. By Proposition 7.7, the total cost of this process is roughly only twice as much as a single run of Algorithm 7.1 since

$$\begin{aligned} & O(C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega+1} \log(\delta\bar{d}) + C^s \mathsf{M}_{\mathbb{K}}(d/2)\, \delta^{\omega+1} \log(\delta\bar{d}) + C^s \mathsf{M}_{\mathbb{K}}(d/4)\, \delta^{\omega+1} \log(\delta\bar{d}) + \cdots) \\ = \;& O(C^s \mathsf{M}_{\mathbb{K}}(d)\, \delta^{\omega+1} \log(\delta\bar{d})). \end{aligned}$$

With these modifications in hand, we ensure that $\mathsf{A}(d) \leqslant 2d$.

PROPOSITION 7.16. *Let $(\mathbb{A}_i)_{i\leqslant t}$ be an explicitly separable tower as in the introduction of this section. If* $\operatorname{card} \mathbb{K} > d^2$ *and $d = n^{O(1)}$, then two $n \times n$ matrices in $\mathbb{A}_t^{n\times n}$ can be multiplied in time $O(n^\omega d)$.*

Proof. Let $\alpha < (\omega-2)/(\varpi-1)$ be a fixed positive constant. Take $\delta = n^\alpha$ and fix a sequence $i_0 < \cdots < i_s$ such that (7.1) and (7.2) hold. Since $d = n^{O(1)}$, we observe that $s = O(1)$.

We first compute a single product in $\mathbb{A}_t$ with the sole purpose of retrieving a parametric splitting $\mathbb{A}_t \cong \mathbb{D}_1 \oplus \cdots \oplus \mathbb{D}_\ell$, together with the auxiliary data for efficient conversions in both directions, and an accelerated tower of height s for each $\mathbb{D}_i$ of degree $\leqslant 2D_i$, where $D_i := \dim_{\mathbb{K}} \mathbb{D}_i$; as discussed above. This precomputation takes time

$$\tilde{O}(C^s d\, \delta^{\varpi+1}) = \tilde{O}(d\, n^{(\varpi+1)\alpha}) = \tilde{O}(d\, n^{\omega-2(\varpi+1-\omega)/(\varpi-1)}) = O(n^\omega d).$$

One conversion between $\mathbb{A}_t^{n\times n}$ and $\mathbb{D}_1^{n\times n} \oplus \cdots \oplus \mathbb{D}_\ell^{n\times n}$ also take time

$$\tilde{O}(C^s d\, \delta^{\varpi-1} n^2) = \tilde{O}(d\, n^{2+\alpha(\omega-2)}) = O(n^\omega d).$$

Now assume that we wish to multiply two matrices $M, N \in \mathbb{A}_t^{n\times n}$. This problem is reduced to ℓ multiplication problems of matrices in $\mathbb{D}_i^{n\times n}$ that we further transform into matrix multiplications within the corresponding accelerated towers, so Proposition 7.15, yields the cost

$$\begin{aligned} & O\left(\sum_{i=1}^{\ell} 2^s n^\omega D_i + 2^s n^2 \mathsf{M}_{\mathbb{K}}(D_i) \log d + C^s n^2 \mathsf{M}_{\mathbb{K}}(D_i) \right) \\ = \;& O(2^s n^\omega d + 2^s n^2 \mathsf{M}_{\mathbb{K}}(d) \log d + C^s n^2 \mathsf{M}_{\mathbb{K}}(d)) \\ = \;& O(n^\omega d + n^2 \mathsf{M}_{\mathbb{K}}(d) \log d) \\ = \;& O(n^\omega d). \end{aligned}$$

using the assumptions that $\omega > 2$ and $d = n^{O(1)}$. □

Remark 7.17. In view of the above proposition, it is likely that determinants can also be computed in time $O(n^\omega d)$. It is an interesting question how to adapt the setting of this paper so that complexity bounds of this type can be derived.

8. CONCLUSION

Theorems 4.2, 5.5, and 7.10 show that the approach of directed evaluation allows us to compute panoramic evaluations in a time that is arbitrarily close to linear. We chose the setting of computation trees as our complexity model and it is natural to ask whether similar results hold in the RAM and Turing models. This seems plausible, although technical difficulties are likely to arise in the efficient implementation of data structures on Turing machines; see [30] for a detailed analysis of a similar kind, but for another problem.

So far, we have only addressed the sequential complexity of dynamic evaluation and its variants. It would be interesting to conduct a similar study for parallel computation models. Dynamic evaluation lends itself particularly well to parallel execution: whenever we encounter a splitting, it is natural to continue the execution of each of the branches in parallel. In the directed setting, we *a priori* have to complete the execution of the principal branch in order to benefit from the fastest possible multi-remaindering. Nevertheless, this is really a trade-off, and we may very well start to execute some "sufficiently thick" residual branches before completion of the principal branch. It is also important to notice that these considerations have only minor impact from the worst case complexity perspective: the opportunities for this kind of parallelism only arise when splittings actually occur. In the absence of splittings, we have to evaluate the computation tree T over the full original algebra $\mathbb{A}$, so the parallel complexity is really a function of the amount of parallelism that is available in T and in the algebra operations of $\mathbb{A}$.

For our presentation, we restricted our towers to be explicitly separable from the outset. In practice, towers are usually constructed by adding new parameters that satisfy polynomial constraints one by one. These polynomials may contain multiple factors, in which case one is usually only interested in the radical ideal generated by the constraints. This more general setting can actually also be covered efficiently using our techniques. Indeed, consider an explicitly separable tower $(\mathbb{A}_i)_{i\leqslant t}$ and a new algebraic parameter α_{t+1} that is subjected to $\mu_{t+1}(\alpha_{t+1}) = 0$, where $\mu_{t+1} \in \mathbb{A}_t[x_{t+1}]$ is a monic polynomial that is not necessarily separable. Then we simply compute the separable factorization $\mu_{t+1} = \varphi_1 \varphi_2^2 \cdots \varphi_n^n$ as if $\mathbb{A}_t$ were a field (here assuming that the characteristic is zero or sufficiently large), using panoramic evaluation, together with the requested Bézout relations for the non-trivial factors $\varphi_i \neq 1$. Let $\mathbb{A}_t \cong \mathbb{D}_1 \oplus \cdots \oplus \mathbb{D}_\ell$ be the resulting splitting. Then any pair $(\varphi_i, \mathbb{D}_j)$ with $\varphi_i \neq 1$ gives rise to a new explicitly separable tower of height $t+1$, which allows us to recurse.

In this paper, we only considered parameters that satisfy polynomial constraints, but the technique of directed evaluation naturally generalizes to various other settings. For instance, for an integer parameter $r \in \mathbb{Z}$, one may compute in $\mathbb{Z}/r\mathbb{Z}$ as if it were a field, so r is partially factored during directed evaluations. This situation is commonly encountered in number theory and computer algebra, where the deterministic construction of "large" prime numbers is rather expensive. Another possible extension of our work concerns real algebraic numbers in the continuation of [16].

Another direction for future work concerns the particular kind of tower arithmetic that we build on. In this paper, we essentially combined the best previously known generic algorithms from Theorem 2.6 with the idea of accelerated towers. For specific base fields $\mathbb{K}$, such as finite fields or subfields of $\mathbb{C}$, other efficient algorithms have been proposed for tower arithmetic [9, 27, 28]. It should not be hard to combine directed evaluation with these algorithms in order to improve on Theorem 7.10 and Corollary 7.12 for such more specific base fields $\mathbb{K}$.

Finally, the recent new theoretical ideas from [27, 28, 29] and this paper should also allow for more efficient software implementations, although it remains a major challenge to make this happen. In particular, this requires to implement all available approaches with care and determine the thresholds for which they are most efficient. For particularly long parametric evaluations, one may also wish to switch to towers of smaller height, and to spend more time on factoring the defining polynomials.

Acknowledgments. We thank the anonymous referees for their useful comments.

BIBLIOGRAPHY

[1] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inform. Process. Lett.*, 18:147–150, 1984.

[2] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and real computation*. Springer Science & Business Media, 2012.

[3] F. Boulier, F. Lemaire, and M. Moreno Maza. Well known theorems on triangular systems and the D5 principle. In J.-G. Dumas, editor, *Proceedings of Transgressive Computing 2006: a conference in honor of Jean Della Dora*, pages 79–92. U. J. Fourier, Grenoble, France, 2006.

[4] P. A. Broadbery, T. Gómez-Díaz, and S. M. Watt. On the implementation of dynamic evaluation. In A. H. M. Levelt, editor, *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation*, ISSAC '95, pages 77–84, New York, NY, USA, 1995. ACM.

[5] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften*. Springer-Verlag, 1997.

[6] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28:693–701, 1991.

[7] M. Coste, H. Lombardi, and M.-F. Roy. Dynamical method in algebra: Effective nullstellensätze. *Ann. Pure Appl. Logic*, 111(3):203–256, 2001.

[8] X. Dahan, M. Moreno Maza, É. Schost, and Yuzhen Xie. On the complexity of the D5 principle. In J.-G. Dumas, editor, *Proceedings of Transgressive Computing 2006: a conference in honor of Jean Della Dora*, pages 149–168. U. J. Fourier, Grenoble, France, 2006.

[9] L. De Feo, J. Doliskani, and É. Schost. Fast algorithms for l-adic towers over finite fields. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, ISSAC '13, pages 165–172, New York, NY, USA, 2013. ACM.

[10] J. Della Dora, C. Dicrescenzo, and D. Duval. About a new method for computing in algebraic number fields. In B. F. Caviness, editor, *EUROCAL '85: European Conference on Computer Algebra Linz, Austria, April 1–3 1985 Proceedings Vol. 2: Research Contributions*, pages 289–290. Springer Berlin Heidelberg, 1985.

[11] S. Dellière. *Triangularisation de systèmes constructibles : application à l'évaluation dynamique*. PhD thesis, Université de Limoges. Faculté des sciences et techniques, 1999.

[12] S. Dellière. On the links between triangular sets and dynamic constructible closure. *J. Pure Appl. Algebra*, 163(1):49–68, 2001.

[13] C. Dicrescenzo and D. Duval. Algebraic extensions and algebraic closure in Scratchpad II. In P. Gianni, editor, *Symbolic and Algebraic Computation. International Symposium ISSAC '88. Rome, Italy, July 4–8, 1988. Proceedings*, number 358 in Lect. Notes Comput. Sci., pages 440–446. Springer-Verlag, 1989.

[14] D. Duval. Rational Puiseux expansions. *Compos. Math.*, 70(2):119–154, 1989.

[15] D. Duval. Algebraic numbers: An example of dynamic evaluation. *J. Symbolic Comput.*, 18(5):429–445, 1994.

[16] D. Duval and L. González-Vega. Dynamic evaluation and real closure. *Math. Comput. Simul.*, 42(4-6):551–560, 1996.

[17] D. Duval and J.-C. Reynaud. Sketches and computation – II: dynamic evaluation and applications. *Math. Structures Comput. Sci.*, 4(2):239–271, 1994.

[18] H. Friedman. Algorithmic procedures, generalized Turing algorithms, and elementary recursion theory. In R. O. Gandy and C. M. E. Yates, editors, *Logic Colloquium '69*, volume 61 of *Studies in Logic and the Foundations of Mathematics*, pages 361–389. Elsevier, 1971.

[19] A. Fröhlich and J. C. Shepherdson. On the factorisation of polynomials in a finite number of steps. *Math. Z.*, 62:331–334, 1955.

[20] A. Fröhlich and J. C. Shepherdson. Effective procedures in field theory. *Philos. Trans. Roy. Soc. London. Ser. A.*, 248:407–432, 1956.

[21] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, New York, 3rd edition, 2013.

[22] T. Gómez-Díaz. Examples of using dynamic constructible closure. *Math. Comput. Simul.*, 42(4-6):375–383, 1996.

[23] D. Harvey and J. van der Hoeven. Faster polynomial multiplication over finite fields using cyclotomic coefficient rings. *J. Complexity*, 54:101404, 2019.

[24] D. Harvey, J. van der Hoeven, and G. Lecerf. Faster polynomial multiplication over finite fields. *J. ACM*, 63(6), 2017. Article 52.

[25] J. van der Hoeven. *Automatic asymptotics*. PhD thesis, École polytechnique, Palaiseau, France, 1997.

[26] J. van der Hoeven and G. Lecerf. Composition modulo powers of polynomials. In M. Blurr, editor, *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 445–452, New York, NY, USA, 2017. ACM.

[27] J. van der Hoeven and G. Lecerf. Modular composition via complex roots. Technical report, CNRS & École polytechnique, 2017. `http://hal.archives-ouvertes.fr/hal-01455731`.

[28] J. van der Hoeven and G. Lecerf. Modular composition via factorization. *J. Complexity*, 48:36–68, 2018.

[29] J. van der Hoeven and G. Lecerf. Accelerated tower arithmetic. *J. Complexity*, 55:101402, 2019.

[30] J. van der Hoeven and G. Lecerf. Fast multivariate multi-point evaluation revisited. *J. Complexity*, 56:101405, 2020.

[31] Xiaohan Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *J. Complexity*, 14(2):257–299, 1998.

[32] E. Kaltofen. On computing determinants of matrices without divisions. In P. S. Wang, editor, *ISSAC 92 International Symposium on Symbolic Algebraic Computation*, ISSAC '92, pages 342–349, New York, NY, USA, 1992. ACM.

[33] K. S. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. *SIAM J.Comput.*, 40(6):1767–1802, 2011.

[34] F. Le Gall. Powers of tensors and fast matrix multiplication. In K. Nabeshima, editor, *ISSAC'14: International Symposium on Symbolic and Algebraic Computation*, pages 296–303, New York, NY, USA, 2014. ACM.

[35] R. Lebreton. Relaxed Hensel lifting of triangular sets. *J. Symbolic Comput.*, 68(2):230–258, 2015.

[36] G. Lecerf. Computing the equidimensional decomposition of an algebraic closed set by means of lifting fibers. *J. Complexity*, 19(4):564–596, 2003.

[37] G. Lecerf. On the complexity of the Lickteig–Roy subresultant algorithm. *J. Symbolic Comput.*, 92:243–268, 2019.

[38] A. Poteaux and É. Schost. Modular composition modulo triangular sets and applications. *Comput. Complex.*, 22(3):463–516, 2013.

[39] A. Poteaux and É. Schost. On the complexity of computing with zero-dimensional triangular sets. *J. Symbolic Comput.*, 50:110–138, 2013.

[40] F. P. Preparata and D. V. Sarwate. An improved parallel processor bound in fast matrix inversion. *Inform. Process. Lett.*, 7(3):148–150, 1978.

[41] J. C. Reynolds. The discoveries of continuations. *LISP and Symbolic Computation*, 6:233–247, 1993.