On the complexity of the D5 principle

Xavier Dahan Marc Moreno Maza Éric Schost Yuzhen Xie

March 17, 2006

Abstract

The D5 Principle was introduced in 1985 by Jean Della Dora, Claire Dicrescenzo and Dominique Duval in their celebrated note "About a new method for computing in algebraic number fields". This innovative approach automatizes reasoning based on case discussion and is also known as "Dynamic Evaluation". Applications of the D5 Principle have been made in Algebra, Computer Algebra, Geometry and Logic.

Many algorithms for solving polynomial systems symbolically need to perform standard operations, such as GCD computations, over coefficient rings that are direct products of fields rather than fields. We show in this paper how asymptotically fast algorithms for polynomials over fields can be adapted to this more general context, thanks to the D5 Principle.

1 Introduction

The standard approach for computing with an algebraic number is through the data of its irreducible minimal polynomial over some base field k. However, in typical tasks such as polynomial system solving, involving many algebraic numbers of high degree, following this approach will require using probably costly factorization algorithms. Jean Della Dora, Claire Dicrescenzo and Dominique Duval introduced "Dynamic Evaluation" techniques (also termed "D5 Principle") as a means to compute with algebraic numbers, while avoiding factorization. Roughly speaking, this approach leads one to compute over *direct products of field extensions of k*, instead of only field extensions.

Applications of Dynamic Evaluation have been made by many authors: González-López and Recio (1993), Gómez Díaz (1994), Duval (1994), Lombardi (2003) and others. Many algorithms for polynomial system solving rely on this philosophy; see, for instance, the work of Lazard (1992), Kalkbrener (1993), Dellière (1999), Moreno Maza (2000), Mora (2003). Boulier et al. (2006).

This work is aiming at filling the lack of complexity results for this approach. The addition and multiplication over a direct product of fields are easily proved to be *quasi-linear* (in a natural complexity measure). As for the inversion, it has to be replaced by *quasi-inversion*: following the D5 philosophy, meeting zero-divisors in the computation will lead to *splitting* the direct product of fields into a family thereof. It is much more tricky to prove quasi-linear complexity estimate for quasi-inversion, because the algorithm relies on other algorithms, for which such an estimate has to be proved: the GCD and the splitting algorithms.

Every *triangular set* T encodes a direct product of fields $\mathbb{K}(T)$ and a *triangular decomposition* of T describes a decomposition of $\mathbb{K}(T)$ into such direct products. These fundamental notions are defined hereafter. In what follows, we assume that the base field k is perfect.

Definition 1.1. A *triangular set* T is a family of n-variate polynomials over k:

$$T \quad = (T_1(X_1)\ ,\ \ T_2(X_1, X_2)\ ,\ \ldots\ ,\ \ T_n(X_1, \ldots, X_n))\,,$$

which forms a reduced Gröbner basis for the lexicographic order induced by $X_n > \cdots > X_1$, and such that the ideal $\langle T \rangle$ generated by T in $k[X_1, \ldots, X_n]$ is radical.

If T is a triangular set, the residue class ring $\mathbb{K}(T) := k[X_1, \ldots, X_n]/\langle T \rangle$ is a direct product of fields. Hence, our questions can be basically rephrased as studying the complexity of operations (addition, multiplication, quasi-inversion) modulo triangular sets. The following notation helps us quantify these algorithms.

Definition 1.2. We denote by $\deg_i(T)$ the degree of T_i in X_i, for all $1 \leq i \leq n$, and by $\deg(T)$ the product $\deg_1(T) \cdots \deg_n(T)$. We call it the *degree* of T.

Observe that $\langle T \rangle$ is zero-dimensional and that for all $1 \leq i \leq n$, the set $(T_1 \ldots, T_i)$ is a triangular set of $k[X_1, \ldots, X_i]$. The zero-set of T in the affine space $\mathbb{A}^n(\bar{k})$ has a particular feature: it is *equiprojectable* (Aubry and Valibouze, 2000; Dahan and Schost, 2004); besides, its cardinality equals $\deg(T)$.

Definition 1.3. A *triangular decomposition* of a zero-dimensional radical ideal $I \subset k[X_1, \ldots, X_n]$ is a family $\mathbf{T} = T^1, \ldots, T^e$ of triangular sets, such that $I = \langle T^1 \rangle \cap \cdots \cap \langle T^e \rangle$ and $\langle T^i \rangle + \langle T^j \rangle = \langle 1 \rangle$ for all $i \neq j$. A triangular decomposition $\mathbf{T}'$ of I *refines* another decomposition $\mathbf{T}$ if for every $T \in \mathbf{T}$ there exists a (necessarily unique) subset $\mathrm{decomp}(T, \mathbf{T}') \subseteq \mathbf{T}'$ which is a triangular decomposition of $\langle T \rangle$.

Let T be a triangular set, let $\mathbf{T} = T^1, \ldots, T^e$ be a triangular decomposition of $\langle T \rangle$, and define $\mathbb{K}(\mathbf{T}) := \mathbb{K}(T^1) \times \cdots \times \mathbb{K}(T^e)$. Then by the Chinese remainder theorem, $\mathbb{K}(T) \simeq \mathbb{K}(\mathbf{T})$. Now let $\mathbf{T}'$ be a refinement of $\mathbf{T}$. For each triangular set T^i in $\mathbf{T}$, denote by $U^{i,1}, \ldots, U^{i,e_i}$ the triangular sets in $\mathrm{decomp}(T^i, \mathbf{T}')$. We have the following e isomorphism:

$$\phi_i :\ \mathbb{K}(T^i) \simeq \mathbb{K}(U^{i,1}) \times \cdots \times \mathbb{K}(U^{i,e_i}), \tag{1}$$

which extend to the following e isomorphisms, where y is a new variable.

$$\Phi_i :\ \mathbb{K}(T^i)[y] \simeq \mathbb{K}(U^{i,1})[y] \times \cdots \times \mathbb{K}(U^{i,e_i})[y]. \tag{2}$$

Definition 1.4. For $\mathbf{h} = (h_1, \ldots, h_e) \in \mathbb{K}(T^1)[y] \times \cdots \times \mathbb{K}(T^e)[y]$, we call *split* of $\mathbf{h}$ with respect to $\mathbf{T}$ and $\mathbf{T}'$, and write $\mathrm{split}(\mathbf{h}, \mathbf{T}, \mathbf{T}')$ the vector $(\Phi_1(h_1), \ldots, \Phi_e(h_e))$.

Note that if $g \in \mathbb{K}(T)[y]$, then we have $\mathrm{split}(g, \{T\}, \mathbf{T}') = \mathrm{split}(\mathrm{split}(g, \{T\}, \mathbf{T}), \mathbf{T}')$. Moreover, we define $\mathrm{split}(g, \mathbf{T}) = \mathrm{split}(g, \{T\}, \mathbf{T})$.

We now introduce a fundamental notion, that of *non-critical* decompositions. It is motivated by the following remark. Let $\mathbf{T} = T^1, \dots, T^e$ be a family of triangular sets, with $T^j = (T_1^j, T_2^j, \dots, T_n^j)$. For $1 \le i \le n$, we write $T_{\le i}^j = T_1^j, T_2^j, \dots, T_i^j$ and define the family $\mathbf{T}_{\le i}$ by:

$$\mathbf{T}_{\le i} = \{T_{\le i}^j \mid j \le e\} \quad \text{(with no repetition allowed).}$$

Even if $\mathbf{T}$ is a triangular decomposition of a 0-dimensional radical ideal $I \subset k[X_1, \dots, X_n]$, $\mathbf{T}_{\le i}$ is not necessarily a triangular decomposition of $I \cap k[X_1, \dots, X_i]$. Indeed, with $n = 2$ and $e = 2$, consider $T^1 = ((X_1 - 1)(X_1 - 2), X_2)$ and $T^2 = ((X_1 - 1)(X_1 - 3), X_2 - 1)$. The family $\mathbf{T} = T^1, T^2$ is a triangular decomposition of the ideal $I = \langle T^1 \rangle \cap \langle T^2 \rangle$. However, the family of triangular sets $\mathbf{T}_{\le 1}$ is not a triangular decomposition since $\langle T_1^1 \rangle + \langle T_1^2 \rangle = \langle X_1 - 1 \rangle$.

Definition 1.5. Let T be a triangular set in $k[X_1, \dots, X_n]$. Two polynomials $a, b \in \mathbb{K}(T)[y]$ are *coprime* if the ideal $\langle a, b \rangle \subset \mathbb{K}(T)[y]$ equals $\langle 1 \rangle$.

Definition 1.6. Let $T \ne T'$ be two triangular sets. The least integer ℓ such that $T_\ell \ne T'_\ell$ is called the *level* of the pair $\{T, T'\}$. The pair $\{T, T'\}$ is *critical* if T_ℓ and T'_ℓ are not relatively prime in $k[X_1, \dots, X_{\ell-1}]/\langle T_1, \dots, T_{\ell-1} \rangle [X_\ell]$. A triangular decomposition $\mathbf{T}$ of $\langle T \rangle$ is *non-critical* if $\mathbf{T}$ has no critical pairs, otherwise it is said to be *critical*.

The pair $\{T^1, T^2\}$ in the above example has level 1 and is critical. Consider $U^{1,1} = (X_1 - 1, X_2)$, $U^{1,2} = (X_1 - 2, X_2)$, $U^{2,1} = (X_1 - 1, X_2 - 1)$ and $U^{2,2} = (X_1 - 3, X_2 - 1)$. Observe that $\mathbf{T}' = \{U^{1,1}, U^{1,2}, U^{2,1}, U^{2,2}\}$ is a non-critical triangular decomposition of I refining $\{T^1, T^2\}$ and that $\mathbf{T}'_{\le 1}$ is a triangular decomposition $I \cap k[X_1, X_2]$.

This notion of critical pair is fundamental. In fact, fast algorithms for the innocuous splitting operations Φ_i of Equation (2) are not guaranteed for critical decompositions, as shown in the following extension of the previous example. Consider a third triangular set $T^3 = ((X_1 - 2)(X_1 - 3), X_2 + X_1 - 3)$. One checks that $\mathbf{U} = \{T^1, T^2, T^3\}$ is a triangular decomposition of $T = ((X_1 - 1)(X_1 - 2)(X_1 - 3), X_2(X_2 - 1))$. However, splitting an element p from $\{T\}$ to $\mathbf{U}$ requires to compute

$$p \bmod (X_1 - 1)(X_1 - 2), \; p \bmod (X_1 - 1)(X_1 - 3), \; p \bmod (X_1 - 2)(X_1 - 3),$$

whence some redundancies. In general, these redundancies prevent the splitting computation from being quasi-linear w.r.t. $\deg(T)$. But if the triangular decomposition is non-critical, then there is no more redundancy, and the complexity of splitting p can be hoped to be quasi-linear.

Removing critical pairs of a critical triangular decomposition in order to be able to split fast requires to delete the common factors between the polynomials involved in the decomposition. To do it fast, (in quasi-linear time) the *coprime factorization*, or *gcd-free basis computation*, algorithm is used. Of course to implement this algorithm over a direct product of fields, one first need to be able to compute GCDs over such a product in quasi-linear time.

Since $\mathbb{K}(T)$ is a direct product of fields, any pair of univariate polynomials $f, g \in \mathbb{K}(T)[y]$ admit a GCD h in $\mathbb{K}(T)[y]$, in the sense that the ideals $\langle f, g\rangle$ and $\langle h\rangle$ coincide, see Moreno Maza and Rioboo (1995). However, even if f, g are both monic, there may not exist a monic polynomial h in $\mathbb{K}(T)[y]$ such that $\langle f, g\rangle = \langle h\rangle$ holds. Consider for instance $f = y + \frac{a+1}{2}$ (assuming that 2 is invertible in k) and $g = y + 1$ where $a \in \mathbb{K}(T)$ satisfies $a^2 = a$, $a \neq 0$ and $a \neq 1$. GCDs with non-invertible leading coefficients are of limited practical interest; this leads us to the following definition.

Definition 1.7. Let f, g be in $\mathbb{K}(T)[y]$. An *extended greatest common divisor (XGCD)* of f and g is a sequence $((h_i, u_i, v_i, T^i), 1 \leq i \leq e)$, where $\mathbf{T} = T^1, \ldots, T^e$ is a non-critical decomposition of T and for all $1 \leq i \leq e$, h_i, u_i, v_i are polynomials in $\mathbb{K}(T^i)[y]$, such that the following holds. Let $f_1, \ldots, f_e = \operatorname{split}(f, \{T\}, \mathbf{T})$ and $g_1, \ldots, g_e = \operatorname{split}(g, \{T\}, \mathbf{T})$; then for $1 \leq i \leq e$, we have:

- h_i is monic or null,
- the inequalities $\deg u_i < \deg g_i$ and $\deg v_i < \deg f_i$ hold,
- the equalities $\langle f_i, g_i\rangle = \langle h_i\rangle$ and $h_i = u_i f_i + v_i g_i$ hold.

One easily checks that such XGCDs exists, and can be computed, for instance by applying the D5 Principle to the Euclidean algorithm. In order to divide f by g in $\mathbb{K}(T)[y]$, we need to check whether the leading coefficient of g is invertible. For this purpose, the following notion is convenient.

Definition 1.8. A *quasi-inverse* of an element $f \in \mathbb{K}(T)$ is a sequence of couples $((u_i, T^i), 1 \leq i \leq e)$ where $\mathbf{T} = T^1, \ldots, T^e$ is a non-critical decomposition of T and u_i is an element of $\mathbb{K}(T^i)$ for all $1 \leq i \leq e$, such that the following holds. Let $f_1, \ldots, f_e = \operatorname{split}(f, \{T\}, \mathbf{T})$; then for $1 \leq i \leq e$ we have either $f_i = u_i = 0$, or $f_i u_i = 1$.

To compute GCDs in quasi-linear time over a direct product of fields, we adapt the *Half-GCD* techniques (Yap, 1993) in Section 4 and explain why its complexity is preserved. This requires a careful inductive process that we summarize in this paper.

- We first need complexity estimates for multiplication modulo a triangular set and splitting w.r.t. triangular decompositions. This is done in Section 3.
- Assuming that multiplications and quasi-inverse computations can be computed fast in $\mathbb{K}(T)$, and assuming fast non-critical refining for triangular decompositions of T, we obtain in Section 4 a fast algorithm for computing GCDs in $\mathbb{K}(T)[y]$. Note that Langemyr (1991) states that GCD's over products of fields can be computed in quasi-linear time, but with no proof.
- Assuming that GCDs can be computed fast in $\mathbb{K}(T_1, \ldots, T_{n-1})[X_n]$, we present fast algorithms for quasi-inverses in $\mathbb{K}(T)$ (Section 5), coprime factorization for polynomials in $\mathbb{K}(T_1, \ldots, T_{n-1})[X_n]$ (Section 6) and refining a triangular decomposition $\mathbf{T}$ of T into a non-critical one (Section 7).

These are the basic blocks for our inductive process, which yields our main results:

Theorem 1.9. *For any $\varepsilon > 0$, there exists $A_\varepsilon > 0$ such that addition, multiplication and quasi-inversion in $\mathbb{K}(T)$ can be computed in $A_\varepsilon^n \deg(T)^{1+\varepsilon}$ operations in k.*

Theorem 1.10. *There exists $G > 0$, and for any $\varepsilon > 0$, there exists $A_\varepsilon > 0$, such that one can compute an extended greatest common divisor of polynomials in $\mathbb{K}(T)[y]$, with degree at most d, using at most $G\, A_\varepsilon^n\, d^{1+\varepsilon} \deg(T)^{1+\varepsilon}$ operations in k.*

Due to space constraints, it is not possible to give all details of our algorithms in this paper. Hence, some algorithms like GCD receive a detailed treatment, while we have to be more sketchy on other ones.

2 Complexity notions

We start by recalling basic results for operations on univariate polynomials.

Definition 2.1. A *multiplication time* is a map $\mathsf{M} : \mathbb{N} \to \mathbb{R}$ such that:

- For any ring R, polynomials of degree less than d in $R[X]$ can be multiplied in at most $\mathsf{M}(d)$ operations $(+, \times)$ in R.
- For any $d \le d'$, the inequalities $\frac{\mathsf{M}(d)}{d} \le \frac{\mathsf{M}(d')}{d'}$ and $\mathsf{M}(dd') \le \mathsf{M}(d)\mathsf{M}(d')$ hold.

Note that in particular the inequalities $\mathsf{M}(d) \ge d$ and $\mathsf{M}(d) + \mathsf{M}(d') \le \mathsf{M}(d + d')$ holds for all d, d'. The following result is due to Cantor and Kaltofen (1991), following the work of Schönhage and Strassen: There exists $c \in \mathbb{R}$ such that the function $d \mapsto c\, d \operatorname{logp}(d) \operatorname{logp} \operatorname{logp}(d)$ is a multiplication time. In what follows, the function logp is defined by $\operatorname{logp}(x) = 2 \log_2(\max\{2, x\})$: this function turns out to be more convenient than the classical logarithm for handling inequalities.

Fast polynomial multiplication is the basis of many other fast algorithms: Euclidean division, computation of the subproduct tree (see Chapter 10 in von zur Gathen and Gerhard (1999)), and multiple remaindering.

Proposition 2.2. *There exists a constant $C \ge 1$ such that the following holds over any ring R. Let M be a multiplication time. Then:*

1. *Dividing in $R[X]$ a polynomial of degree less than $2d$ by a monic polynomial of degree at most d requires at most $5\mathsf{M}(d) + O(d) \le C\, \mathsf{M}(d)$ operations $(+, \times)$ in R.*
2. *Let F be a monic polynomial of degree d in $R[X]$. Then additions and multiplications in $R[X]/F$ requires at most $6\, \mathsf{M}(d) + O(d) \le C\, \mathsf{M}(d)$ operations $(+, \times)$ in R.*
3. *Let $F_1, \dots, F_s$ be non-constant monic polynomials in $R[X]$, with sum of degrees d. Then one can compute the subproduct tree associated to $F_1, \dots, F_s$ using at most $\mathsf{M}(d) \operatorname{logp}(d)$ operations $(+, \times)$ in R.*

4. *Let $F_1, \ldots, F_s$ be non-constant monic polynomials in $R[X]$, with sum of degrees d. Then given A in $R[X]$ of degree less than d, one can compute $A \bmod F_1, \ldots, A \bmod F_s$ within $11\,\mathsf{M}(d)\operatorname{logp}(d) + O(d\operatorname{logp}(d)) \le C\,\mathsf{M}(d)\operatorname{logp}(d)$ operations $(+, \times)$ in R.*

5. *Assume that R is a field. Then, given two polynomials in $R[X]$ of degree at most d, computing their monic GCD and their Bézout coefficients can be done in no more than $33\,\mathsf{M}(d)\operatorname{logp}(d) + O(d\operatorname{logp}(d)) \le C\,\mathsf{M}(d)\operatorname{logp}(d)$ operations $(+, \times, /)$ in R.*

6. *Assume that R is a field and that F is a monic squarefree polynomial in $R[X]$ of degree d. Then, computing a quasi-inverse modulo F of a polynomial $G \in R[X]$ of degree less than d can be done in no more than $71\,\mathsf{M}(d)\operatorname{logp}(d) + O(d\operatorname{logp}(d)) \le C\,\mathsf{M}(d)\operatorname{logp}(d)$ operations $(+, \times, /)$ in R.*

Proof. The first point is proved in Theorem 9.6 of (von zur Gathen and Gerhard, 1999) and implies the second one. The third and fourth points are proved in Lemma 10.4 and Theorem 10.15 of the same book. The fifth point is reported in Theorem 11.5 of that book (with a better constant), and is a particular case of Section 4 of this article. If F has no multiple factors in $R[X]$, a quasi-inverse of G modulo F can be obtained by at most two extended GCD computations and one division with entries of degree at most d. Using estimates for the GCD leads to the result claimed in point 6. □

We now define our key complexity notion, arithmetic time for triangular sets.

Definition 2.3. An *arithmetic time* is a function $T \mapsto \mathsf{A}_n(T)$ with real positive values and defined over all triangular sets in $k[X_1, \ldots, X_n]$ such that the following conditions hold

(E_0) For every triangular decomposition $\mathbf{T} = T^1, \ldots, T^e$ of T, we have $\mathsf{A}_n(T^1) + \cdots + \mathsf{A}_n(T^e) \le \mathsf{A}_n(T)$.

(E_1) Every addition or multiplication in $\mathbb{K}(T)$ can be done in at most $\mathsf{A}_n(T)$ operations in k.

(E_2) Every quasi-inverse in $\mathbb{K}(T)$ can be computed in at most $\mathsf{A}_n(T)$ operations in k.

(E_3) Given a triangular decomposition $\mathbf{T}$ of T, one can compute a *non-critical* triangular decomposition $\mathbf{T}'$ which refines $\mathbf{T}$, in at most $\mathsf{A}_n(T)$ operations in k.

(E_4) For every $\alpha \in \mathbb{K}(T)$ and every non-critical triangular decomposition $\mathbf{T}$ of T, one can compute $\operatorname{split}(\alpha, \{T\}, \mathbf{T})$ in at most $\mathsf{A}_n(T)$ operations in k.

Our main goal in this paper is then to give estimates for arithmetic times. This is done through an inductive proof; the following proposition gives such a result for the base case, triangular sets in one variable.

Proposition 2.4. *If $n = 1$, then $T \in k[X_1] \mapsto C\,\mathsf{M}(\deg T)\operatorname{logp}(\deg T)$ is an arithmetic time.*

Proof. A triangular set in one variable is simply a squarefree monic polynomial in $k[X_1]$. Hence, (E_1), (E_2) and (E_4) respectively follow from points 2, 6 and 4 in Proposition 2.2. Property (E_0) is clear. Since $n = 1$, all triangular decompositions are non-critical, and (E_3) follows. $\square$

3 Basic complexity results: multiplication and splitting

This section is devoted to give first complexity results for triangular sets: we give upper bounds on the cost of multiplication, and splitting. In general, we do not know how to perform this last operation in quasi-linear time; however, when the decomposition is non-critical, quasi-linearity can be reached.

Proposition 3.1. *Let M be a multiplication function, and let C be the constant from Proposition 2.2. Let T be a triangular set in $k[X_1, \dots, X_n]$. Then:*

- *Additions and multiplications modulo T can be done in at most $C^n \prod_{i \le n} \mathsf{M}(\deg_i T)$ operations in k.*
- *If $\mathbf{T}$ is a non-critical decomposition of T, then for any h in $\mathbb{K}(T)$, $\mathrm{split}(h, \{T\}, \mathbf{T})$ can be computed in at most $n\, C^n \prod_{i \le n} \mathsf{M}(\deg_i T) \,\mathrm{logp}(\deg_i T)$ operations in k.*

Proof. The first part of the proposition is easy to deal with: the case of additions is obvious, using the inequality $\mathsf{M}(d) \ge d$; as to multiplication, an easy induction using point (1) in Proposition 2.2 gives the result. The end of the proof uses point (4) in Proposition 2.2; the non-critical assumption is then used through the following lemma. $\square$

Lemma 3.2. *Consider a non-critical decomposition $\mathbf{T}$ of the triangular set $T = (T_1, \dots, T_n)$. Write $\mathbf{T}_{\le n-1} = \{U^1, \dots, U^s\}$, and, for all $i \le s$, denote by $T^{i,1}, \dots, T^{i,e_i}$ the triangular sets in $\mathbf{T}$ such that $T^{i,j} \cap k[X_1, \dots, X_{n-1}] = U^i$ (thus $\mathbf{T}$ is the set of all $T^{i,j}$, with $i \le s$ and $j \le e_i$). Then $\mathbf{T}_{\le n-1}$ is a non-critical decomposition of the triangular set $(T_1, \dots, T_{n-1})$. Moreover, for all $i \le s$, we have:*

$$\sum_{j \le e_i} \deg_n T^{i,j} = \deg_n T.$$

As an illustration, consider again, for $n = 2$, the triangular sets

$$\begin{array}{rcl} T^1 & = & ((X_1-1)(X_1-2),\ X_2) \\ T^2 & = & ((X_1-1)(X_1-3),\ X_2-1) \\ \text{and } T^3 & = & ((X_1-2)(X_1-3),\ X_2+X_1-3). \end{array}$$

These triangular sets form a critical decomposition $\mathbf{T}$ of the ideal $\langle T^1 \rangle \cap \langle T^2 \rangle \cap \langle T^3 \rangle$, which is also generated by $T = ((X_1-1)(X_1-2)(X_1-3), X_2(X_2-1))$.

Here, $\mathbf{T}_{\le 1}$ is given by $\{U^1, U^2, U^3\} = \{(X_1-1)(X_1-2), (X_1-1)(X_1-3), (X_1-1)(X_1-3)\}$, so that $s = 3$. Take for instance $U^1 = (X_1-1)(X_1-2)$; then we have $e_1 = 1$ and $T^{1,e_1} = T^1$. Note then that $\deg_2 T^{1,e_1} = 1$ differs from $\deg_2 T = 2$, so the conclusion of the previous lemma is indeed violated.

4 Fast GCD computations modulo a triangular set

GCDs of univariate polynomials over a field can be computed in quasi-linear time by means of the *Half-GCD* algorithm (Brent et al., 1980; Yap, 1993). We show how to adapt this technique over the direct product of fields $\mathbb{K}(T)$ and how to preserve its complexity class. Throughout this section, we consider $T \mapsto \mathsf{A}_n(T)$ an arithmetic time for triangular sets in $k[X_1, \ldots, X_n]$.

Proposition 4.1. *For all $a, b \in \mathbb{K}(T)[y]$ with $\deg a, \deg b \leq d$, one can compute an extended greatest common divisor of a and b in $O(\mathsf{M}(d)\log(d))\mathsf{A}_n(T)$ operations in k.*

We prove this result by describing our GCD algorithm over the direct product of fields $\mathbb{K}(T)$ and its complexity estimate. We start with two auxiliary algorithms.

Monic forms. Any polynomial over field can be made monic by division through its leading coefficient. Over a product of fields, this division may induce splittings. We now study this issue.

Definition 4.2. A *monic form* of $f \in \mathbb{K}(T)[y]$ is a sequence of quadruples $((u_i, v_i, m_i, T_i), 1 \leq i \leq e)$, where $\mathbf{T} = T^1, \ldots, T^e$ is a non-critical decomposition of T, u_i, v_i are in $\mathbb{K}(T^i)$ and m_i is in $\mathbb{K}(T^i)[y]$ for all $1 \leq i \leq e$, and such that the following holds.

Let $f_1, \ldots, f_e = \text{split}(f, \{T\}, \mathbf{T})$. Denote by $\text{lc}(f_i)$ the leading coefficient of f_i. Then, for all $1 \leq i \leq e$ we have $u_i = \text{lc}(f_i)$, and $m_i = v_i f_i$, and either $u_i = v_i = 0$ or $u_i v_i = 1$.

Observe that for all $1 \leq i \leq e$, the polynomial m_i is monic or null.

The following algorithm shows how to compute a monic form. This function uses a procedure quasiInverse$(\mathbf{f}, \mathbf{T})$. This procedure takes as input a triangular decomposition $\mathbf{T} = T^1, \ldots, T^e$ of T and a sequence $\mathbf{f} = f_1, \ldots, f_e$ in $\mathbb{K}(T^1)[y] \times \cdots \times \mathbb{K}(T^e)[y]$ and returns a sequence $(((f_{ij}, T^{ij}), 1 \leq j \leq e_i), 1 \leq i \leq e)$ where $((f_{ij}, T^{ij}), 1 \leq j \leq e_i)$ is a quasi-inverse of f_i modulo T^i and such that $(T^{ij}, 1 \leq j \leq e_i, 1 \leq i \leq e)$ is a non-critical refinement of $\mathbf{T}$. Its complexity is studied in Section 5.

The number at the end of a line, multiplied by $\mathsf{A}_n(T)$, gives an upper bound for the total time spent at this line. Therefore, the following algorithm computes a monic form of f in at most $(8d + 6)\mathsf{A}_n(T)$ operations in k.

```
monic($f, T$) ==
  1     $\mathbf{T} := \{T\}$
  2     $\mathbf{v} := (0)$
  3     $g := f$
  4     while $g \neq 0$ repeat
  4.1      $\mathbf{u} := \text{split}(\text{lc}(g), \{T\}, \mathbf{T})$                    $[d+1]$
  4.2      $(\mathbf{w}, \mathbf{T}') := \text{quasiInverse}(\mathbf{u}, \mathbf{T})$             $[3d+3]$
  4.3      $\mathbf{v} := \text{split}(\mathbf{v}, \mathbf{T}, \mathbf{T}')$                       $[d+1]$
  4.4      for $1 \leq i \leq \#\mathbf{v}$ repeat
```

```
4.4.1        if $v_i = 0$ then $v_i := w_i$                    $[d+1]$
4.5      $\mathbf{T} := \mathbf{T}'$
4.6      $g := g - \mathrm{leadingTerm}(g)$
5    $\mathbf{f} := \mathrm{split}(f, \{T\}, \mathbf{T})$                    $[d]$
6    $\mathbf{u} := \mathrm{lc}(\mathbf{f})$
7    $\mathbf{m} := \mathbf{v} \cdot \mathbf{f}$                    $[d]$
8    return $((u_i, v_i, m_i, T^i), 1 \le i \le \#\mathbf{T})$
```

Division with monic remainder. The previous notion can then be used to compute Euclidean divisions, producing *monic* remainders: they will be required in our fast Euclidean algorithm for XGCDs.

Definition 4.3. Let $f, g \in \mathbb{K}(T)[y]$ with g monic. A *division with monic remainder* of f by g is a sequence of tuples $((g_i, q_i, v_i, u_i, r_i, T^i), 1 \le i \le e)$ such that $\mathbf{T} = T^1, \dots, T^e$ is a non-critical decomposition of T, and, for all $1 \le i \le e$, we have $u_i, v_i \in \mathbb{K}(T^i)$ and $g_i, q_i, r_i, \in \mathbb{K}(T^i)[y]$, and such that the following holds.

Let $f_1, \dots, f_e = \mathrm{split}(f, \{T\}, \mathbf{T})$ and $g_1, \dots, g_e = \mathrm{split}(g, \{T\}, \mathbf{T})$. Then, for all $1 \le i \le e$, the polynomial r_i is null or monic, we have either $u_i = v_i = 0$ or $u_i v_i = 1$, and the polynomials q_i and $u_i r_i$ are the quotient and remainder of f_i by g_i in $\mathbb{K}(T^i)[y]$.

The following algorithm computes a division with monic remainder of f by g and requires at most $(5\mathsf{M}(d) + O(d))\mathsf{A}_n(T)$ operations in k. We write $(q, r) = \mathrm{div}(f, g)$ for the quotient and the remainder in the (standard) division with remainder in $\mathbb{K}(T)[y]$.

```
mdiv$(f, g, T)$ ==
  1    $(q, r) := \mathrm{div}(f, g)$                    $[5\mathsf{M}(d) + O(d)]$
  2    $((u_i, v_i, r_i, T^i), 1 \le i \le \#\mathbf{T}) := \mathrm{monic}(r, T)$                    $[8d - 2]$
  3    $(q_i, 1 \le i \le \#\mathbf{T}) := \mathrm{split}(q, \{T\}, \mathbf{T})$                    $[d+1]$
  4    $(g_i, 1 \le i \le \#\mathbf{T}) := \mathrm{split}(g, \{T\}, \mathbf{T})$                    $[d]$
  5    return $((g_i, q_i, u_i, v_i, T^i), 1 \le i \le \#\mathbf{T})$
```

We are now ready to generalize the *Half-Gcd* method as exposed in Yap (1993). We introduce the following operations. For $a, b \in \mathbb{K}(T)[y]$ with $0 < \deg b < \deg a = d$, each of the algorithms $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$ and $\mathrm{M}_{\mathrm{hgcd}}(a, b, T)$ returns a sequence $((M_1, T^1), \dots, (M_e, T^e))$ where

(s_1) $\mathbf{T} = T^1, \dots, T^e$ is a non-critical triangular decomposition of T,

(s_2) M_i is a square matrix of order 2 with coefficients in $\mathbb{K}(T^i)[y]$,

such that, if we define $(a_1, \dots, a_e) = \mathrm{split}(a, \{T\}, \mathbf{T})$ and $(b_1, \dots, b_e) = \mathrm{split}(b, \{T\}, \mathbf{T})$, then, for all $1 \le i \le e$, defining $(t_i, s_i) = (a_i, b_i)\ {}^t M_i$, we have

(s_3) in the case of $\mathrm{M}_{\mathrm{gcd}}$, the polynomial t_i is a GCD of a_i, b_i and $s_i = 0$ holds,

(s'_3) in the case of $\mathrm{M}_{\mathrm{hgcd}}$, the ideals $\langle t_i, s_i \rangle$ and $\langle a_i, b_i \rangle$ of $\mathbb{K}(T^i)[y]$ are identical, and $\deg s_i < \lceil d/2 \rceil \leq \deg t_i$ holds.

The algorithm below implements $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$, and is an extension of the analogue algorithm known over fields. Observe that if the input triangular set T is not decomposed during the algorithm, in particular if $\mathbb{K}(T)$ is a field, then the algorithm yields generators of the ideal $\langle a, b \rangle$. If T is decomposed, then the lines from 5 to 7.3.1 guarantee that $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$ generates a non-critical triangular decomposition of T.

```
$\mathrm{M}_{\mathrm{gcd}}(a,b,T)$ ==
  0        $\mathbf{G} := [\,]$; $\mathbf{T} := [\,]$;
  1        $((M_i, T^i), 1 \leq i \leq e) := \mathrm{M}_{\mathrm{hgcd}}(a, b, T)$                      $[H(d)]$
  2        $(a_1, \ldots, a_e) := \mathrm{split}(a, (T^i, 1 \leq i \leq e))$                     $[O(d)]$
  3        $(b_1, \ldots, b_e) := \mathrm{split}(b, (T^i, 1 \leq i \leq e))$                     $[O(d)]$
  4        for $i$ in $1 \cdots e$ repeat
  4.1          $(t_i, s_i) := (a_i, b_i)\ {}^t M_i$                                   $[4\,\mathsf{M}(d) + O(d)]$
  4.2          if $s_i = 0$ then
  4.2.1            $\mathbf{G} := \mathbf{G}, (M_i, T^i)$
  4.2.2            $\mathbf{T} := \mathbf{T}, T^i$
  4.3          $((s_{ij}, q_{ij}, r_{ij}, u_{ij}, v_{ij}, T^{ij}), 1 \leq j \leq e_i) := \mathrm{mdiv}(t_i, s_i)$  $[\frac{5}{2}\mathsf{M}(d) + O(d)]$
  4.4          $(M_{ij}, 1 \leq j \leq e_i) := \mathrm{split}(M_i, (T^{ij}, 1 \leq j \leq e_i))$            $[O(d)]$
  4.5          for $j$ in $1 \cdots e_i$ repeat
  4.5.1            $M_{ij} := \begin{pmatrix} 0 & 1 \\ v_{ij} & -q_{ij} v_{ij} \end{pmatrix} M_{ij}$                      $[2\,\mathsf{M}(d) + O(d)]$
  4.5.2            if $r_{ij} = 0$ then
  4.5.2.1              $\mathbf{G} := \mathbf{G}, (M_{ij}, T^i)$
  4.5.2.2              $\mathbf{T} := \mathbf{T}, T^{ij}$
  4.5.3            $((N_{ijk}, T^{ijk}), 1 \leq k \leq e_{ij}) := \mathrm{M}_{\mathrm{gcd}}(s_{ij}, r_{ij}, T^{ij})$           $[G(d/2)]$
  4.5.4            $(M_{ijk}, 1 \leq k \leq e_{ij}) := \mathrm{split}(M_{ij}, (T^{ijk}, 1 \leq k \leq e_{ij}))$       $[O(d)]$
  4.5.5            for $k$ in $1 \cdots e_{ij}$ repeat
  4.5.5.1              $M_{ijk} := N_{ijk} M_{ijk}$                               $[8\,\mathsf{M}(d) + O(d)]$
  4.5.5.2              $\mathbf{G} := \mathbf{G}, (M_{ijk}, T^{ijk})$
  4.5.5.3              $\mathbf{T} := \mathbf{T}, T^{ijk}$
  5        $\mathbf{T}' := \mathrm{removeCriticalPairs}(\mathbf{T})$                                  [1]
  6        $\mathbf{Res} := [\,]$
  7        for $(M, T) \in \mathbf{G}$ repeat
  7.1          $\mathsf{U} := \mathrm{decomp}(T, \mathbf{T}')$
  7.2          $(M_\ell, 1 \leq \ell \leq \#\mathsf{U}) := \mathrm{split}(M, \{T\}, \mathsf{U})$                  $[O(d)]$
  7.3          for $1 \leq \ell \leq \#\mathsf{U}$ do
  7.3.1                $\mathbf{Res} := \mathbf{Res}, (M_i, U^i)$
  8        return $\mathbf{Res}$
```

The Half-GCD algorithm can be adapted to $\mathbb{K}(T)[y]$ (not reported here due to space consideration) leading to an implementation of $\mathrm{M}_{\mathrm{hgcd}}(a, b, T)$. It has a structure very similar to $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$, see (Yap, 1993) for details in the case the coefficients lie in a field.

Now, we give running time estimates for $\mathrm{M}_{\mathrm{hgcd}}(a, b, T)$ and $\mathrm{M}_{\mathrm{gcd}}(a, b, T)$. For $0 < \deg b < \deg a = d$, we denote by $G(d)$ and $H(d)$ respective upper bounds for the running time of $\mathrm{M}_{\mathrm{gcd}}(a, b)$ and $\mathrm{M}_{\mathrm{hgcd}}(a, b)$, in the sense that both operations can be done in respective times $G(d)\mathsf{A}_n(T)$ and $H(d)\mathsf{A}_n(T)$.

The number at the end of an above line, multiplied by $\mathsf{A}_n(T)$, gives an upper bound of the running time of this line. These estimates follow from the super-linearity of the arithmetic time for triangular sets, the running time estimates of the operation $\mathrm{mdiv}(f, g, T)$ and classical degree bounds for the intermediate polynomials in the Extended Euclidean Algorithms; see for instance Chapter 3 in (von zur Gathen and Gerhard, 1999). Therefore, counting precisely the degrees appearing, we have: $G(d) \leq G(d/2) + H(d) + (33/2)\mathsf{M}(d) + O(d)$. The operation $\mathrm{M}_{\mathrm{hgcd}}(a, b, T)$ makes two recursive calls with input polynomials of degree at most $d/2$, leading to $H(d) \leq 2H(d/2) + (33/2)\mathsf{M}(d) + O(d)$. The superlinearity of M implies

$$H(d) \leq \frac{33}{2}\mathsf{M}(d)\,\log d + O(d\,\log\,d) \quad \text{and} \quad G(d) \leq 2H(d) + 2\mathsf{M}(d) + O(d).$$

This leads to the result reported in Proposition 4.1.

We conclude with a specification of a function used in the remaining sections. For a triangular decomposition $\mathbf{T} = T^1, \ldots, T^e$ of T, two sequences $\mathbf{f} = f_1, \ldots, f_e$ and $\mathbf{g} = g_1, \ldots, g_e$ of polynomials in $\mathbb{K}(T^1)[y], \ldots, \mathbb{K}(T^e)[y]$, the operation $\mathrm{xgcd}(\mathbf{f}, \mathbf{g}, \mathbf{T})$ returns a sequence $(((g_{ij}, u_{ij}, v_{ij}, T^{ij}), 1 \leq j \leq e_i), 1 \leq i \leq e)$ where $((g_{ij}, u_{ij}, v_{ij}, T^{ij}), 1 \leq j \leq e_i)$ is an extended greatest common divisor of f_i and g_i and such that $(T^{ij}, 1 \leq j \leq e_i, 1 \leq i \leq e)$ is a non-critical refinement of $\mathbf{T}$.

Proposition 4.1 implies that if $f_1, \ldots, f_e, g_1, \ldots, g_e$ have degree at most d then $\mathrm{xgcd}(\mathbf{f}, \mathbf{g}, \mathbf{T})$ runs in at most $O(\mathsf{M}(d)\log(d))\mathsf{A}_n(T)$ operations in k.

5 Fast computation of quasi-inverses

Throughout this section, we consider an arithmetic time A_{n-1} for triangular sets in $n-1$ variables. We explain how a quasi-inverse can be computed fast with the algorithms *split*, *xgcd*, and *removeCriticalPairs*.

Proposition 5.1. *Let $T = (T_1, \ldots, T_n)$ be a triangular set with $\deg_i(T) = d_i$ for all $1 \leq i \leq n$. Let f be in $\mathbb{K}(T)$. Then one can compute a quasi-inverse of f modulo T in $O\big(\mathsf{M}(d_n)\log(d_n)\big)\mathsf{A}_{n-1}(T_{<n})$ operations in k.*

We consider first the case where f is a non-constant polynomial and its degree w.r.t. X_n is positive and less than d_n. We give the algorithm, followed by the necessary explanations. Here, the quantity at the end a line, once multiplied by $\mathsf{A}_{n-1}(T_{<n})$, gives the total amount of time spent at this line. Af the end of this section, we briefly discuss the other cases to be considered for f.

```
quasiInverse_n$(f, T)$ ==
  1        $((g_i, u_i, v_i, T^i_{<n}), 1 \leq i \leq e) := \mathrm{xgcd}(f, T_n, T_{<n})$          $\big[O\big(\mathsf{M}(d_n)\log(d_n)\big)\big]$
  2        $(T^i_n, \ldots, T^e_n) := \mathrm{split}(T_n, \{T_{<n}\}, \{T^1_{<n}, \ldots, T^e_{<n}\})$          $[O(d_n)]$
  3        $(f_i, \ldots, f_e) := \mathrm{split}(f, \{T_{<n}\}, \{T^1_{<n}, \ldots, T^e_{<n}\})$          $[O(d_n)]$

  4        $\mathbf{T} := \{\}$;   $\mathbf{C} := \{\}$;   **result** $:= \{\}$;
  5        **for** $i = 1 \ldots e$ **do**
  5.1         **if** $\deg(g_i) = 0$ **then**
  5.1.1          $\mathbf{C} := \mathbf{C},\ (u_i, T^i_{<n} \cup T^i_n)$;   $\mathbf{T} := \mathbf{T},\ T^i_{<n} \cup T^i_n$
  5.2         **else if** $\deg(g_i) > 0$ **then**
  5.2.1          $\mathbf{C} := \mathbf{C},\ (0, T^i_{<n} \cup g_i)$;   $\mathbf{T} := \mathbf{T},\ T^i_{<n} \cup g_i$
  5.2.2          $q_i := \mathrm{quotient}(T^i_n, g_i)$          $[5\mathsf{M}(d_n) + O(d_n)]$
  5.2.3          $((g_{ij}, u_{ij}, v_{ij}, T^{ij}_{<n}), 1 \leq j \leq e_i) := \mathrm{xgcd}(f_i, q_i, T^i_{<n})$
  5.2.4          $(T^{i1}_n, \ldots, T^{ie_i}_n) := \mathrm{split}(q_i, \{T^i_{<n}\}, \{T^{i1}_{<n}, \ldots, T^{ie_i}_{<n}\})$          $[O(d_n)]$
  5.2.5          **for** $j = 1 \ldots e_i$ **do**
  5.2.5.1           $\mathbf{C} := \mathbf{C},\ (u_{ij}, T^{ij}_{<n} \cup T^{ij}_n)$;   $\mathbf{T} := \mathbf{T},\ T^{ij}_{<n} \cup T^{ij}_n$
  6        $\mathbf{T'_{<n}} := \mathrm{removeCriticalPairs}(\mathbf{T_{<n}})$          $O(1)$
  7        **for** $(u, S) \in \mathbf{C}$ **do**
  7.1         $(R^1, \ldots, R^l) := \mathrm{decomp}(S_{<n}, \mathbf{T'_{<n}})$
  7.2         $(S^1_n, \ldots, S^l_n) := \mathrm{split}(S_n, \{S_{<n}\}, \{R^1, \ldots, R^l\})$          $[O(d_n)]$
  7.3         $(u_1, \ldots, u_l) := \mathrm{split}(u, \{S_{<n}\}, \{R^1, \ldots, R^l\})$          $[O(d_n)]$
  7.4         **result** := **result**, $((u_k, R^k \cup S^k_n), 1 \leq k \leq l)$
  8        **return result**
```

We first calculate an extended greatest common divisor of f and T_n modulo the triangular set $T_{<n} = (T_1, \ldots, T_{n-1})$. This induces a non-critical decomposition $\{T^1_{<n}, \ldots, T^e_{<n}\}$ of $T_{<n}$. For further operations, we compute the images of T_n and f over this decomposition.

Let $1 \leq i \leq e$. If the value of g_i is 1, then u_i is the inverse of f modulo $\{T^i_{<n} \cup T^i_n\}$. Otherwise, $\deg g_i > 0$, and the computation needs to be split into two branches.

In one branch, at line 5.2.1, we build the triangular set $\{T^i_{<n} \cup g_i\}$, modulo which f reduces to zero. In the other branch, starting from line 5.2.2, we build the triangular set as $\{T^i_{<n} \cup q_i\}$, modulo which f is invertible. Indeed since the triangular set $\{T^i_{<n} \cup q_i\}$ generates a radical ideal, T^i_n is squarefree modulo $\{T^i_{<n}\}$, and $\gcd(f, q_i)$ must be 1 modulo $\{T^i_{<n} \cup q_i\}$. Therefore we can simply use the *xgcd* (step 5.2.3) once to compute the quasi-inverse of f modulo $\{T^i_{<n} \cup q_i\}$.

After collecting all the quasi-inverses, we remove the critical pairs in the new family of triangular sets. Since no critical pairs are created at level n in the previous computation, the removal of critical pairs needs only to perform below level n. At the end, we split the inverses and the top polynomials w.r.t the last non-critical decomposition.

We also need quasi-inverse computations in two other different situations. One is when f may not have the same main variable as the triangular set T. We need also to compute the quasi-inverses in the sense of quasiInverse$(\mathbf{f}, \mathbf{T})$ introduced in Section 4 where $\mathbf{T} = T^1, \ldots, T^e$ is a triangular decomposition of T, and $\mathbf{f} = f_1, \ldots, f_e$ is a sequence of polynomials in $k[X_1, \ldots, X_n]$. They are simply built on top of the quasiInverse$_\mathrm{n}(f, T)$, with additional splits and removal of critical pairs.

The dominant cost is the two xgcd calls. Therefore, in each situation, the total cost is bounded by $O\big(\mathsf{M}(d_n)\log(d_n)\big)\mathsf{A}_{n-1}(T_{<n})$.

6 Coprime factorization

We present in this section a quasi-linear time algorithm for coprime factorization of univariate polynomials over a field. Other fast algorithms for this problem are given by (Gautier and Roch, 1997), with a concern for parallel efficiency, and in (Bernstein, 2005), in a wider setting, but with a slightly worse computation time.

Due to space consideration, we present our algorithm only for polynomials over a field k; however, it adapts over a direct product of fields, following the ideas presented in Section 4. We will use this tool in Section 7 for computing non-critical refinements of a triangular decomposition (see the example in the introduction for a motivation of this idea).

Definition 6.1. Let $A = a_1, \ldots, a_s$ be squarefree polynomials in $k[x]$. Some polynomials $b_1, \ldots, b_t$ in $k[x]$ are a *gcd-free basis* of the set A if $\gcd(b_i, b_j) = 1$ for $i \neq j$, each a_i can be written (necessarily uniquely) as a product of some of the b_j, and each b_j divides one of the a_i. The associated *coprime factorization* of A consists in the factorization of all polynomials a_i in terms of the polynomials $b_1, \ldots, b_t$.

Proposition 6.2. *Let d be the sum of the degrees of $A = a_1, \ldots, a_s$. Then a coprime factorization of A can be computed in $O(\mathsf{M}(d)\,\mathrm{logp}(d)^3)$ operations in k.*

For brevity's sake, we will only prove how to compute a gcd-free basis of A, assuming without loss of generality that all a_i have positive degree. Deducing the coprime factorization of A involves some additional bookkeeping operations, keeping track of divisibility relations; it induces no new arithmetic operations, and thus has no consequence on complexity.

The subproduct tree. The subproduct tree is a useful construction to devise fast algorithms with univariate polynomials, in particular the coprime factorization. We review this notion briefly and refer to (von zur Gathen and Gerhard, 1999) for more details.

Let $m_1, \ldots, m_r$ be monic, non-constant, polynomials in $k[x]$. The subproduct tree Sub associated to $m_1, \ldots, m_r$ is defined as follows. If $r = 1$, then Sub is a single node, labeled by the polynomial m_1. Else, let $r' = \lceil r/2 \rceil$, and let Sub_1 and Sub_2 be the trees associated to $m_1, \ldots, m_{r'}$ and $m_{r'+1}, \ldots, m_r$ respectively. Let p_1 and p_2 be the polynomials at the roots of Sub_1 and Sub_2. Then Sub is the tree whose root is labeled by the product $p_1 p_2$ and has children Sub_1 and Sub_2. A *row* of the tree consists in all nodes lying at some given distance from the root. The *depth* of the tree is the number of its non-empty rows. Let $d = \sum_{i=1}^{r} \deg(m_i)$; then the sum of the degrees of the polynomials on any row of the tree is at most d, and its depth is at most $\mathrm{logp}(d)$.

We now define some subroutines required for our gcd-free basis algorithm, starting by the computation of multiple gcd's. Recall that the cost at given any line in our pseudo-code denotes the total time spent at this line; for simplicity, in what follows, we omit the $O(\ )$ in the complexity estimates attached to the pseudo-code.

Multiple gcd's. The first algorithm takes as input p and $(a_1, \dots, a_e)$ in $k[x]$, and outputs the sequence of all $\gcd(p, a_i)$. The idea of this algorithm is to first reduce p modulo all a_i using fast simultaneous reduction, and then take the gcd's of all remainders with the polynomials a_i (see also Exercise 11.4 in (von zur Gathen and Gerhard, 1999)).

We make the assumption that all a_i are non-constant in the pseudo-code below, so as to apply the results of Proposition 2.2. To cover the general case, it suffices to introduce a wrapper function, that strips the input sequence $(a_1, \dots, a_e)$ from its constant entries, and produces 1 as corresponding gcd's; this function induces no additional arithmetic cost. Finally, we write $d = \sum_{i=1}^{e} \deg a_i$.

```
multiGcd($p, (a_1, \dots, a_e)$) ==
  1   if $\deg p \geq d$ then $p := p \bmod (a_1 \cdots a_e)$          $[\mathsf{M}(\deg p) + \mathsf{M}(d)\,\mathrm{logp}(d)]$
  2   $(q_1, \dots, q_e) := (p \bmod a_1, \dots, p \bmod a_e)$          $[\mathsf{M}(d)\,\mathrm{logp}(d)]$
  3   return $(\gcd(q_1, a_1), \dots, \gcd(q_e, a_e))$          $\left[\sum_i \mathsf{M}(\deg a_i)\,\mathrm{logp}(\deg a_i)\right]$
```

The cost of lines 1 and 2 follows from Proposition 2.2. The function $d \mapsto \mathsf{M}(d)\,\mathrm{logp}(d)$ is super-additive, so the complexity at line 3 fits in $O(\mathsf{M}(d)\,\mathrm{logp}(d))$. Hence, the total cost of this algorithm is in $O(\mathsf{M}(\deg p) + \mathsf{M}(d)\,\mathrm{logp}(d))$.

Pairs of gcd's. The next step is to compute several pairs of gcd's. On input, we take two families of polynomials $(a_1, \dots, a_e)$ and $(b_1, \dots, b_s)$, where all a_i (resp. all b_i) are squarefree and pairwise coprime. The following algorithm computes all $\gcd(a_i, b_j)$. As above, we suppose that all a_i are non-constant; to cover the general case, it suffices to introduce a wrapper function, with arithmetic cost 0, that removes each constant a_i from the input, and adds the appropriate sequence $(1, \dots, 1)$ in the output. Here, we write $d = \max(\sum_i \deg a_i, \sum_j \deg b_j)$.

```
pairsOfGcd($(a_1, \dots, a_e), (b_1, \dots, b_s)$) ==
  1       Build a subproduct tree $\mathsf{Sub}(a_1, \dots, a_e)$ and let $f = \mathrm{RootOf}(\mathsf{Sub})$      $[\mathsf{M}(d)\,\mathrm{logp}(d)]$
  2       Label the root of $\mathsf{Sub}$ by multiGcd($f, \{b_1, \dots, b_s\}$)      $[\mathsf{M}(d)\,\mathrm{logp}(d)]$
  3       for every node $N \in \mathsf{Sub}$, going top-down do
  3.1         if $N$ is not a leaf and has label $\mathbf{g}$ then
  3.1.1           $f_1 :=$ leftChild($N$); $f_2 :=$ rightChild($N$);
  3.1.2           Label $f_1$ by multiGcd($f_1, \mathbf{g}$)      $[\mathsf{M}(d)\,\mathrm{logp}(d)^2]$
  3.1.3           Label $f_2$ by multiGcd($f_2, \mathbf{g}$)      $[\mathsf{M}(d)\,\mathrm{logp}(d)^2]$
```

This algorithm computes the gcd's of $(b_1, \dots, b_s)$ with all polynomials in the subproduct tree associated with $(a_1, \dots, a_e)$; the requested output can be found at the leaves of the tree. To give the complexity of this algorithm, one proves that the total number of operations along each row is in $O(\mathsf{M}(d)\,\mathrm{logp}(d))$, whence a total cost in $O(\mathsf{M}(d)\,\mathrm{logp}(d)^2)$.

A special case of gcd-free basis. The input of our third subroutine are sequences of polynomials $(a_1, \ldots, a_e)$ and $(b_1, \ldots, b_s)$, where all a_i (resp. all b_i) are squarefree and pairwise coprime. We compute a gcd-free basis of $(a_1, \ldots, a_e, b_1, \ldots, b_s)$; this is done by computing all $\gcd(a_i, b_j)$, as well as the quotients $\delta_i = a_i / \prod_j \gcd(a_i, b_j)$ and $\gamma_j = b_j / \prod_i \gcd(a_i, b_j)$.

We denote by removeConstants(L) a subroutine that removes all constant polynomials from a sequence L (such a function requires no arithmetic operation, so its cost is zero in our model). In the complexity analysis, we still write $d = \max(\sum_i \deg a_i, \sum_j \deg b_j)$.

```
gcdFreeBasisSpecialCase($(a_1, \ldots, a_e), (b_1, \ldots, b_s)$) ==
  1      $(g_{i,j})_{1 \le i \le e, 1 \le j \le s}$ := pairsOfGcd($(a_1, \ldots, a_e), (b_1, \ldots, b_s)$)      $[\mathsf{M}(d)\,\mathrm{logp}(d)^2]$
  2      for $j$ in $1 \ldots s$ do
  2.1        $L_j$ := removeConstants($g_{1,j}, \ldots, g_{e,j}$)
  2.2        $\beta_j := \prod_{\ell \in L_j} \ell$      $[\mathsf{M}(d)\,\mathrm{logp}(d)]$
  2.3        $\gamma_j := b_j$ quo $\beta_j$      $[\mathsf{M}(d)]$
  3      for $i$ in $1 \ldots e$ do
  3.1        $L_i$ := removeConstants($g_{i,1}, \ldots, g_{i,s}$)
  3.2        $\alpha_i := \prod_{\ell \in L_i} \ell$      $[\mathsf{M}(d)\,\mathrm{logp}(d)]$
  3.3        $\delta_i := a_i$ quo $\alpha_i$      $[\mathsf{M}(d)]$
  4      return removeConstants($g_{1,1}, \ldots, g_{i,j}, \ldots, g_{e,s}, \gamma_1, \ldots, \gamma_s, \delta_1, \ldots, \delta_e$)
```

The validity of this algorithm is easily checked. The estimates for the cost of lines 2.2, 2.3, 3.2 and 3.3 come for the cost necessary to build a subproduct tree and perform Euclidean division, together with the fact that β_j (resp. α_i) divides b_j (resp. a_i). The total cost is thus in $O(\mathsf{M}(d)\,\mathrm{logp}(d)^2)$.

Gcd-free basis. We can finally give our algorithm for gcd-free basis. As input, we take squarefree, non-constant polynomials $a_1, \ldots, a_e$, and write $d = \sum_{i \le e} \deg a_i$. We need a construction close to the subproduct tree: we form a binary tree Sub' whose nodes will be labeled by sequences of polynomials. Initially the leaves contain the sequences of length 1 $(a_1), \ldots, (a_e)$, and all other nodes are empty. Then, we go up the tree; at a node N, we use the previous subroutine to compute a gcd-free basis of the sequences labeling the children of N.

```
gcdFreeBasis($\{a_1, \ldots, a_e\}$) ==
  1        Build the tree $\mathsf{Sub}'(a_1, \ldots, a_e)$
  2        for every node $N \in \mathsf{Sub}'$ and from bottom-up repeat
  2.1          if $N$ is not a leaf then
  2.1.1            $f_1$ := leftChild($N$) ; $f_2$ := rightChild($N$)
  2.1.2            Label $N$ by gcdFreeBasisSpecialCase($f_1, f_2$)      $[\mathsf{M}(d)\,\mathrm{logp}(d)^3]$
  3        return the label of RootOf($\mathsf{Sub}'$)
```

The total number of operations at a node N of the subset tree is $O(\mathsf{M}(d_N)\,\mathrm{logp}(d_N)^2)$, where d_N is sum of the degrees of the polynomials lying at the two children of N. Summing over all nodes, using the tree structure, the total cost is seen to be in $O(\mathsf{M}(d)\,\mathrm{logp}(d)^3)$ operations, as claimed.

7 Removing critical pairs

We next show how to remove critical pairs. This is an inductive process, whose complexity is estimated in the following proposition and its corollary.

We need to extend the notion of "refining" introduced previously. Extending Definition 1.3, we say that a family of triangular sets $\mathbf{T}'$ refines another family $\mathbf{T}$ if for every $T \in \mathbf{T}$, there exists a subset of $\mathbf{T}'$ that forms a triangular decomposition of $\langle T \rangle$. Note the difference with the initial definition: we do not impose that the family $\mathbf{T}$ forms a triangular decomposition of some ideal I. In particular, the triangular sets in $\mathbf{T}$ do not have to generate coprime ideals.

Proposition 7.1. *There exists a constant K such that the following holds. Let $\mathsf{A}_1, \ldots, \mathsf{A}_{n-1}$ be arithmetic times for triangular sets in $1, \ldots, n-1$ variables.*

Let T be a triangular set in n variables, and let $\mathbf{U}$ be a triangular decomposition of $\langle T \rangle$. Then for all $j = 1, \ldots, n$, the following holds: given $\mathbf{U}_{\leq j}$, one can compute a non-critical triangular decomposition $\mathbf{W}$ of $T_{\leq j}$ that refines $\mathbf{U}_{\leq j}$ using a_j operations in k, where a_j satisfies the recurrence inequalities $a_0 = 0$ and for $j = 0, \ldots, n-1$,

$$a_{j+1} \leq 2a_j + K\mathsf{M}(d_{j+1} \cdots d_n) \operatorname{logp}(d_{j+1} \cdots d_n)^3 \mathsf{A}_j(T_{\leq j}),$$

and where $d_j = \deg_j T$ for $j = 1, \ldots, n$.

Before discussing the proof of this assertion, let us give an immediate corollary, which follows by a direct induction.

Corollary 7.2. *Given a triangular decomposition $\mathbf{U}$ of $\langle T \rangle$, one can compute a non-critical triangular decomposition $\mathbf{W}$ of $\langle T \rangle$ that refines $\mathbf{U}$ in time*

$$K\left(2^{n-1}\mathsf{M}(d_1 \cdots d_n) \operatorname{logp}(d_1 \cdots d_n)^3 + \cdots + \mathsf{M}(d_n) \operatorname{logp}(d_n)^3 \mathsf{A}_{n-1}(T_{\leq n-1})\right).$$

Proof. We only sketch the proof of the proposition. Let thus j be in $0, \ldots, n-1$ and let $\mathbf{U} = U^1, \ldots, U^e$ be a triangular decomposition of $\langle T \rangle$; we aim at removing the critical pairs in $\mathbf{U}_{\leq j+1}$. Let $\mathbf{V}$ be obtained by removing the critical pairs in $\mathbf{U}_{\leq j}$. Thus, $\mathbf{V}$ consists in triangular sets in $k[X_1, \ldots, X_j]$, and has no critical pair.

Let us fix $i \leq e$, and write $U^i = (U^i_1, \ldots, U^i_n)$. By definition, there exists a subset $\mathbf{V}_i = V^{i,1}, \ldots, V^{i,e_i}$ of $\mathbf{V}$ which forms a non-critical decomposition of $(U^i_1, \ldots, U^i_j)$. Our next step is to compute

$$U^{i,1}_{j+1}, \ldots, U^{i,e_i}_{j+1} = \operatorname{split}(U^i_{j+1}, (U^i_1, \ldots, U^i_j), \mathbf{V}_i).$$

Consider now a triangular set V in $\mathbf{V}$. There may be several subsets $\mathbf{V}_i$ such that $V \in \mathbf{V}_i$. Let $S_V \subset \{1, \ldots, e\}$ be the set of corresponding indices; thus, for any $i \in S_V$, there exists $\ell(i)$ in $1, \ldots, e_i$ such that $V = V^{i,e_{\ell(i)}}$. We will then compute a coprime factorization of all polynomials $U^{i,e_{\ell(i)}}_{j+1}$ in $\mathbb{K}(V)[X_{j+1}]$, for $i \in S_V$, and for all V.

This process will refine the family $\mathbf{V}$, creating possibly new critical pairs: we get rid of these critical pairs, obtaining a decomposition $\mathbf{W}$. It finally suffices to split all polynomials in the coprime factorization obtained before from $\mathbf{V}$ to $\mathbf{W}$ to conclude. The cost estimates then takes into account the cost for the two calls to the same process in j variables, hence the term $2a_j$, and the cost for coprime factorization and splitting. Studying the degrees of the polynomials involved, this cost can be bounded by

$$K\mathsf{M}(d_{j+1}\cdots d_n)\,\mathrm{logp}(d_{j+1}\cdots d_n)^3\mathsf{A}_j(T_{\leq j})$$

for some constant K, according to the results in the last section. □

8 Concluding the proof

All ingredients are now present to give the proof of the following result, which readily implies the main theorems stated in the introduction.

Theorem 8.1. *There exists a constant L such that, writing*

$$\mathsf{A}_n(d_1,\dots,d_n) = L^n \prod_{i\leq n} \mathsf{M}(d_i)\,\mathrm{logp}(d_i)^3,$$

the function $T \mapsto \mathsf{A}_n(\deg_1 T,\dots,\deg_n T)$ is an arithmetic time for triangular sets in n variables, for all n.

Proof. The proof requires to check that taking L big enough, all conditions defining arithmetic times are satisfied. We do it by induction on n; the case $n = 1$ is settled by Proposition 2.4, taking L larger than the constant C in that proposition, and using the fact that $\mathrm{logp}(x) \geq 1$ for all x.

Let us now consider index n; we can thus assume that the function A_j is an arithmetic time for triangular sets in j variables, for $j = 1,\dots,n-1$. Then, at index n, condition (E_0) makes no difficulty, using the super-additivity of the function M. Addition and multiplication (condition (E_1)) and splitting (condition (E_4)) follow from Proposition 3.1, again as soon as the condition $L \geq C$ holds. The computation of quasi-inverses (condition (E_2)) is taken care of by Proposition 5.1, using our induction assumption on A, as soon as L is large enough to compensate the constant factor hidden in the $O(\ )$ estimate of that proposition.

The cost for removing critical pairs is given in the previous section. In view of Corollary 7.2, and using the condition $\mathsf{M}(dd') \leq \mathsf{M}(d)\mathsf{M}(d')$, after a few simplifications, to satisfy condition (E_3), L must satisfy the inequality

$$K(2^{n-1} + 2^{n-2}L + \cdots + L^{n-1}) \leq L^n,$$

where K is the constant introduced in Corollary 7.2. This is the case for L large enough: $L \geq K + 2$ suffices. □

References

Aubry, P., Valibouze, A., 2000. Using Galois ideals for computing relative resolvents. Journal of Symbolic Computation 30 (6), 635–651.

Bernstein, D. J., 2005. Factoring into coprimes in essentially linear time. Journal of Algorithms 54 (1), 1–30.

Boulier, F., Lemaire, F., Moreno Maza, M., 2006. Well known theorems on triangular systems and the D5 Principle. In: TC'2006. University of Granada, Spain.

Brent, R., Gustavson, F., Yun, D., 1980. Fast solution of Toeplitz systems of equations and computations of Padé approximants. Journal of Algorithms 1, 259–295.

Cantor, D., Kaltofen, E., 1991. On fast multiplication of polynomials over arbitrary algebras. Acta Informatica 28, 693–701.

Dahan, X., Moreno Maza, M., Schost, É., Wu, W., Xie, Y., 2005. Lifting techniques for triangular decomposition. In: ISSAC'05. ACM press, pp. 108–115.

Dahan, X., Schost, É., 2004. Sharp estimates for triangular sets. In: ISSAC'04. ACM Press, pp. 103–110.

Della Dora, J., Discrescenzo, C., Duval, D., 1985. About a new method method for computing in algebraic number fields. In EUROCAL 85 Vol. 2. Vol. 204 of *LNCS*. Springer-Verlag.

Dellière, S., 1999. Triangularisation de systèmes constructibles. Application à l'évaluation dynamique. Ph.D. Thesis, Université de Limoges.

Duval, D., 1994. Algebraic numbers: an example of dynamic evaluation. Journal of Symbolic Computation 18 (5), 429–446.

von zur Gathen, J., Gerhard, J., 1999. Modern Computer Algebra. Cambridge University Press.

Gautier, T., Roch, J.-L., 1997. NC2 computation of gcd-free basis and application to parallel algebraic numbers computation. In: PASCO '97. ACM Press, pp. 31–37.

Gómez Díaz, T., 1994. Quelques applications de l'évaluation dynamique. Ph.D. Thesis, Université de Limoges.

González-López, M., Recio, T., 1993. The ROMIN inverse geometric model and the dynamic evaluation method. In: Cohen, A. M. (Ed.), Proc. of the 1991 SCAFI Seminar, Computer Algebra in Industry. Wiley.

Kalkbrener, M., 1993. A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. Journal of Symbolic Computation 15 (2), 143–167.

Langemyr, L., 1991. Algorithms for a multiple algebraic extension. In: Effective methods in algebraic geometry (Castiglioncello, 1990). Birkhäuser Boston, pp. 235–248.

Lazard, D., 1992. Solving zero-dimensional algebraic systems. Journal of Symbolic Computation 13 (2), 117–132.

Lombardi, H., 2006. Structures algébriques dynamiques, espaces topologiques sans points et programme de Hilbert, Annals of Pure and Applied Logic 137, 256–290.

Mora, T., 2003. Solving Polynomial Equation Systems I. The Kronecker-Duval Philosophy. No. 88 in Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Moreno Maza, M., 2000. On triangular decompositions of algebraic varieties. Tech. Rep. 4/99, NAG, UK, Presented at the MEGA-2000 Conference, Bath, UK, http://www.csd.uwo.ca/∼moreno.

Moreno Maza, M., Rioboo, R., 1995. Polynomial gcd computations over towers of algebraic extensions. In: AAECC-11. Vol. 948 of *LNCS*, Springer-Verlag, pp. 365–382.

Yap, C., 1993. Fundamental Problems in Algorithmic Algebra. Princeton University Press.

Xavier Dahan
LIX, École polytechnique 91128 Palaiseau, France
dahan@lix.polytechnique.fr

Marc Moreno Maza
ORCCA, University of Western Ontario (UWO) London, Ontario, Canada
moreno@orcca.on.ca

Éric Schost
LIX, École polytechnique 91128 Palaiseau, France
schost@lix.polytechnique.fr

Yuzhen Xie
ORCCA, University of Western Ontario (UWO) London, Ontario, Canada
yxie@orcca.on.ca