An Algebraic Approach to MaxCSP

Ilario Bonacina
UPC Universitat Politècnica de Catalunya, Spain

Jordi Levy
IIIA, CSIC, Spain

Abstract

Recently, there have been some attempts to base SAT and MaxSAT solvers on calculi beyond Resolution, even trying to solve SAT using MaxSAT proof systems. One of these directions was to perform MaxSAT sound inferences using polynomials over finite fields, extending the proof system Polynomial Calculus, which is known to be more powerful than Resolution.

In this work, we extend the use of the Polynomial Calculus for optimization, showing its completeness over many-valued variables. This allows using a more direct and efficient encoding of CSP problems (e.g., k-Coloring) and solving the maximization version of the problem on such encoding (e.g., Max-k-Coloring).

2012 ACM Subject Classification Theory of computation → Proof complexity; Computing methodologies → Algebraic algorithms; Mathematics of computing → Solvers

Keywords and phrases MaxCSP, Polynomial Calculus, MaxSAT

Digital Object Identifier 10.4230/LIPIcs.SAT.2025.8

Funding This work was supported by the grant numbers PID2022-138506NB-C21, and PID2022-138506NB-C22 funded by AEI.

1 Introduction

The *Constraint Satisfaction Problem* (CSP) is the problem of, given a set of constraints on variables, finding an assignment satisfying all of them. SAT can be seen as the restriction of this problem to Boolean, *i.e.,* 0/1-valued, variables and constraints given as clauses. MaxCSP and MaxSAT are their respective optimization versions, where we find assignments maximizing the number of satisfied restrictions. CSP could be tackled via translations to SAT, see for instance, [21]. MaxCSP could be tackled via translations to MaxSAT, but this kind of reduction is more subtle since we have to preserve not only satisfiability, but also the number of violated constraints. As a result, in some problems, it may be more efficient to try to solve the original problem using CSP techniques directly, instead of these translations.

In this paper, we consider the natural adaptation of the notions of MaxCSP and MaxSAT to sets of polynomials and we show how to use an algebraic framework to tackle optimization problems on sets of polynomials over *multi-valued* variables. Sets of polynomials over many-valued variables can be used to encode CSP and MaxCSP more directly, avoiding the translation to SAT.

Given a set of polynomials P, we want to determine the maximum number of polynomials in P that can be simultaneously evaluated to zero under a common assignment of the variables. Unlike SAT and MaxSAT that rely on Boolean encodings, using sets of polynomials allows to use a direct multi-valued encoding of MaxCSP problems that are more expressive, concise, and closer in nature to possible underlying algebraic properties of the original problem at hand.

The algebraic framework we use is an adaptation of *Polynomial Calculus* (PC) introduced in [9]. PC is the proof system underlying the reasoning power of algorithms to compute Gröbner bases [7, 6, 12], and, from a theoretical point of view, PC is well-studied in connection

with big open questions in Complexity Theory. From a practical point of view, the interest in PC stems from the fact that it is stronger and more versatile than Resolution, the proof system underlying the Conflict-Driven-Clause-Learning paradigm of state-of-the-art SAT-solvers [20, 1]. Moreover, algorithms computing Gröbner bases are useful in practice in the context of *coloring of graphs* [18, 10, 11] and *verification of multiplier circuits* [16, 14, 15, 13]. The previous applications use polynomials over *multi-valued* or *Fourier* encoding, *i.e.,* using variables such that $x^\kappa = 1$. Unlike the case of PC over *Boolean* variables, PC over multi-valued variables is not so well understood. Some well-known principles, such as Tseitin formulas have short (polynomial size) proofs in PC over ± 1-variables, while they do not have short proofs over PC with Boolean variables. Recently, it has also been shown that there are unsatisfiable formulas having polynomial size refutation in PC over Boolean variables, while requiring exponential size in PC over ± 1-valued [19].

Recently, Bonacina, Bonet and Levy [2, 3] introduced an algebraic calculus (wPC_κ) for certifying the optimum value in optimization problems on polynomials over the finite field $\mathbb{F}_\kappa$. This is a generalization of PC to the context of optimization and not only satisfiability. The authors proved the completeness of the calculus in the case of polynomials *over Boolean variables*.

In this work, we consider polynomials over non-Boolean encodings and therefore we give a theoretical ground for extending the applicability of the algorithms using algebraic reasoning and many-valued variables from the context of CSPs to MaxCSPs. For the sake of clarity, we omit the possible generalization to the case of weights and the presence of hard constraints, for instance coming from a Fourier encoding (i.e. using polynomials $x^\ell - 1$ to limit the possible values taken), although all the content of this paper will adapt easily to those cases.

The main contribution of this work is the completeness of the calculus wPC_κ over arbitrary sets of polynomials in $\mathbb{F}_\kappa$ (Theorem 5.1). This argument loosely follows the structure of the completeness of wPC_κ over polynomials forced to take Boolean values in [3] but requires several non-trivial adaptations. This gives an algorithmic procedure that is efficient in some cases, see Section 6.

As a proof of concept, we also show how to apply the calculus wPC_3 to certify Max-3-colorability of graphs exploiting the use of non-Boolean variables (see Section 4).

Structure of the Paper

Section 2 contains all the necessary preliminaries. Section 3 gives the definition of the algebraic framework for optimization and the definition of wPC_κ. Section 4 exemplifies the algebraic framework to Max-coloring. Section 5 contains the completeness of the calculus wPC_κ (Theorem 5.1). This is the main technical contribution of this work. Section 6 gives an application of the algorithmic procedure giving the completeness of wPC_κ. Section 7 give respectively some technical and more general final remarks.

2 Preliminaries

For a natural number $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. Capital letters typically denote sets and multi-sets, for instance, X usually denotes a set of variables. With p, q, f we typically denote polynomials. κ will always denote the number of elements of a generic finite field.

Basic Algebra Notations

Informally, a *field* $\mathbb{F}$ is a set equipped with addition and multiplication that behave similarly as they would do in the rationals or real numbers.

For instance, $\mathbb{F}_3$ is the finite field with 3 elements $\{0, 1, -1\}$. The addition is modulo 3, that is $1 + 1 = -1$ and $-1 - 1 = 1$, the multiplication is the usual one, and for each $a \in \mathbb{F}_3$, $a^3 = a$. For κ a power of a prime, $\mathbb{F}_\kappa$ is the (unique up-to-isomorphism) finite field with κ elements. For every element $a \in \mathbb{F}_\kappa$ it holds that $a^\kappa = a$.

Given a set of variables X and a field $\mathbb{F}$, $\mathbb{F}[X]$ is the ring of polynomials with coefficients in $\mathbb{F}$ and variables in X.

A *assignment* on the variables X is a map $\alpha : X \to \mathbb{F}$. The *evaluation* of a polynomial $p \in \mathbb{F}[X]$ on α is $p(\alpha)$ and it is the element of $\mathbb{F}$ given by substituting each variable x in p with $\alpha(x)$ and then simplifying the resulting expression using the sum and multiplication rules of $\mathbb{F}$. An assignment α *satisfies* a polynomial p if p vanishes on α, i.e., $p(\alpha) = 0$.

MaxSAT on Polynomials

Similarly to the usual MaxSAT, in the optimization problem on polynomials, we are interested in *maximizing* the number of polynomials set to zero on a common assignment. For a multi-set of polynomials $P \subseteq \mathbb{F}[X]$ and an assignment α let

$$\mathrm{cost}(P, \alpha) = |\{p \in P \ : \ p(\alpha) \neq 0\}| \ ,$$

i.e., $\mathrm{cost}(P, \alpha)$ is the number of polynomials in P non vanishing in α (the polynomials are counted with the respective multiplicity in P). The optimization problem on polynomials becomes *minimizing* the quantity $\mathrm{cost}(P, \alpha)$ over all possible assignments α:

$$\mathrm{cost}(P) = \min_\alpha \mathrm{cost}(P, \alpha) \ .$$

In this work we are interested in formal systems that can be used to certify lower bounds on $\mathrm{cost}(P)$, *i.e.,* $\mathrm{cost}(P) \geqslant s$ for some natural number s. To do so we use the notion of *strongly sound* inference rules introduced in the context of MaxSAT Resolution in [4, 5], adapted here from the context of clauses to generic polynomials.

▶ **Definition 2.1** (strongly sound rule). *Let $p_1, \ldots, p_m, q_1, \ldots, q_\ell$ be polynomials in $\mathbb{F}[X]$ and $\dfrac{p_1, \ldots, p_m}{q_1, \ldots, q_\ell}$ be an inference rule. The rule is* strongly sound *if for every assignment $\alpha : X \to \mathbb{F}$,*

$$\mathrm{cost}(\{p_1, \ldots, p_m\}, \alpha) = \mathrm{cost}(\{q_1, \ldots, q_\ell\}, \alpha) \ .$$

The interest in strongly sound rules lies in the fact they can be used to build inference systems for MaxSAT or, in our case, inference systems for optimization on polynomials. This is based on the fact that a strongly sound rule $\dfrac{p_1, \ldots, p_m}{q_1, \ldots, q_\ell}$ on a multi-set of polynomials P with $p_1, \ldots, p_m \in P$ can be used as a *substitution rule* giving the multi-set

$$P' = P \setminus \{p_1, \ldots, p_m\} \cup \{q_1, \ldots, q_\ell\} \ ,$$

where $\mathrm{cost}(P) = \mathrm{cost}(P')$. Hopefully, P' is simpler, and by repeating the process we would like to arrive to a P' which trivially certifies that $\mathrm{cost}(P') \geqslant s$, for instance, since there are s-many constant non-zero polynomials in P'. To certify that $\mathrm{cost}(P) = s$ we additionally need to prove that the remaining non-constant polynomials in P' have a common zero.

Examples of such systems are wPC_3 (Definition 3.1) or, working with clauses instead of polynomials, MaxSAT-Resolution [17, 4, 5].

In this work, we focus on polynomials in $\mathbb{F}[X]$ and actually we are only interested in how they behave *after* being evaluated, therefore it makes sense to consider them modulo the following equivalence relation.

▶ **Definition 2.2** ($\equiv$)**.** *Given $p, q \in \mathbb{F}[X]$, $p \equiv q$ iff for every assignment $\alpha : X \to \mathbb{F}$, $p(\alpha) = q(\alpha)$.*

For every polynomial $p \in \mathbb{F}_\kappa[X]$, it holds that $p^\kappa \equiv p$ and, if $p_1 \equiv q_1$ and $p_2 \equiv q_2$, then $p_1 p_2 \equiv q_1 q_2$. This gives the following strongly sound substitution rule:

$$\frac{p}{q} \quad (\text{EQUIVALENCE RULE}, \ \equiv) \ , \tag{1}$$

where $p \equiv q$. By the Nullstellensatz, it is possible to check efficiently whether $p \equiv q$ since this is equivalent to the polynomial $p - q$ being in the ideal generated by $\{x^\kappa - x \ : x \in X\}$. This latter condition is efficiently checkable "multilinearizing" the polynomial $p - q$, that is substituting each occurrence of variables $x^{a(\kappa-1)+b}$ with x^b, where $b \in \{0, \ldots, \kappa - 1\}$, and then checking that after this process all the terms cancel-out and the polynomial is identically 0.

3 Polynomial Calculus for MaxSAT

Bonacina, Bonet and Levy in [3] introduced *Polynomial Calculus for* MaxSAT (wPC_κ) as a way to use a system stronger than MaxSAT-Resolution to solve MaxSAT. For simplicity, we omit the treatment of weighted polynomials and sets of polynomials as "hard" constraints. We also omit the case of negative weights and therefore what we call wPC_κ is $\mathsf{wPC}_{\mathbb{N},\mathbb{F}_\kappa}$ in [3].

▶ **Definition 3.1** (wPC_κ, [3])**.** *Given a multi-set of polynomials $P \subset \mathbb{F}_\kappa[X]$, a wPC_κ derivation of a polynomial $p \in \mathbb{F}_\kappa[X]$ is a sequence of multi-sets $L_0, \ldots, L_t$ such that*

1. *$L_0 = P$, $p \in L_t$, and*
2. *for each $i > 0$, L_i is the result of the application of a SUM / PROD / $\equiv$ rule as a substitution rule on L_{i-1}, where the SUM and PROD rules are the following:*

$$\frac{p \quad q}{p+q \quad pq \quad p((p+q)^{\kappa-1} - 1)} \ (\textit{SUM}) \qquad \frac{p}{pq \quad p(q^{\kappa-1} - 1)} \ (\textit{PROD}) \tag{2}$$

The size *of a wPC_κ derivation is the total number of symbols in it.*

The system wPC_κ can be used to certify that for a multi-set of polynomials $P \subset \mathbb{F}_\kappa[X]$, $\text{cost}(P) \geqslant s$. Indeed, a wPC_κ derivation from P of s copies of the constant polynomial 1 implies that $\text{cost}(P) \geqslant s$. Or, equivalently, a wPC_κ derivation of s non-zero constant polynomials, since by the PROD rule from any non-zero constant we can derive the polynomial 1 multiplying by its inverse (for an example of application, see last inference step in Example 3.3). In wPC_κ it is also possible to certify that $\text{cost}(P) = s$: if on top of the s-many polynomials identically 1 we give an assignment which is a common zero of the polynomials obtained alongside the 1s.

To clarify the rules for wPC_κ, we give an equivalent, but more explicit, set of rules for $\kappa = 3$:

$$\frac{p \quad q}{p+q \quad pq \quad pq(q-p)} \ (\text{SUM}) \qquad \frac{p}{pq \quad p(q^2-1)} \ (\text{PROD}) \tag{3}$$

where $p, q \in \mathbb{F}_3[X]$. The rule PROD in (3) is strongly sound essentially because for any assignment α exactly one among $q(\alpha)$ and $q(\alpha)^2 - 1$ is zero. The rule SUM in (3) is strongly sound by case analysis: the less trivial case to check is when we have an assignment α such that $\mathrm{cost}(\{p, q\}, \alpha) = 2$, i.e. $p(\alpha) \in \{1, -1\}$ and $q(\alpha) \in \{1, -1\}$. In this case, the cost of the conclusions of the rule is also 2 because $p(\alpha)q(\alpha) \neq 0$ and, over the assignments αs we are considering, exactly one among $p(\alpha) + q(\alpha)$ and $q(\alpha) - p(\alpha)$ is zero.

▶ Remark 3.2 (On the coefficients of the polynomials). Polynomial Calculus on polynomials with coefficients in a generic ring (for instance $\mathbb{Z}$) was also investigated in [8]. Both the article [3] and ours only consider the MaxSAT/MaxCSP adaptation of Polynomial Calculus over finite fields. The reason is that the rules SUM and PROD do not immediately adapt to arbitrary finite rings such as $\mathbb{Z}/_{4\mathbb{Z}}$ (the set of integers with arithmetic modulo 4) or fields of characteristic 0. In particular, the SUM and PROD rules on polynomials with coefficients in $\mathbb{Z}/_{4\mathbb{Z}}$ are *not* strongly sound. This seems to be related to the presence of 0-divisors, e.g. $2 \cdot 2 = 0$ in $\mathbb{Z}/_{4\mathbb{Z}}$. Further investigation is needed to define a suitable version of Polynomial Calculus for MaxSAT or MaxCSP over infinite fields or generic rings.

To illustrate the calculus wPC_3 and the use of the rules from (3) and (1) we give a couple of examples.

▶ **Example 3.3.** Consider the set of polynomials $P = \{x,\ -y,\ -x + y - 1\} \subset \mathbb{F}_3[x, y]$. The following wPC_3 derivation shows that $\mathrm{cost}(P) = 1$ (we use the $\equiv$-rule implicitly):

$$\dfrac{\dfrac{\dfrac{x \quad -y \quad -x+y-1}{-y \quad y-1 \quad -x^2+xy-x \quad xy^2+xy}\,\text{SUM}}{-1 \quad -y^2+y \quad 0 \quad -x^2+xy-x \quad xy^2+xy}\,\text{SUM}}{1 \quad 0 \quad -y^2+y \quad 0 \quad -x^2+xy-x \quad xy^2+xy}\,\text{PROD}$$

The first SUM is between x and $-x + y - 1$. The second SUM is between $-y$ and $y - 1$. The polynomial $xy^2 + xy$ in the first SUM comes from the fact that

$$x \cdot (-x + y - 1) \cdot ((-x + y - 1) - x) \equiv xy^2 + xy\ .$$

The 0 in the second SUM comes from the fact that $(-y) \cdot (y - 1) \cdot ((y - 1) - (-y)) \equiv 0$.

This shows that $\mathrm{cost}(P) \geqslant 1$. To conclude that $\mathrm{cost}(P) = 1$, it is enough to show that the assignment $x = 0$, $y = 0$ is a zero of all polynomials of the next to last multi-set except the polynomial -1. The last step to transform this non-zero constant polynomial into 1 may be omitted. ◇

▶ **Example 3.4.** Consider $P = \{x^2+1\} \subset \mathbb{F}_3[x]$. Since $(x^2+1)^2 = x^4+2x^2+1 \equiv 3x^2+1 \equiv 1$, the following wPC_3 derivation shows that $\mathrm{cost}(P) = 1$:

$$\dfrac{\dfrac{x^2+1}{(x^2+1)\cdot(x^2+1) \quad (x^2+1)\cdot((x^2+1)^2-1)}\,\text{PROD}}{1 \quad 0}\,\equiv$$

That is, in particular, the polynomial x^2+1 doesn't have roots in $\mathbb{F}_3$ (as expected). Proposition 5.5 generalizes this example. ◇

▶ **Example 3.5.** Consider $P = \{-\prod_{i=1}^{n} x_i,\ \prod_{i=1}^{n} x_i - 1\} \subset \mathbb{F}_3[x_1, \ldots, x_n]$. Clearly, P is unsatisfiable and a single application of the SUM rule gives immediately the polynomial 1. On the other hand, encoding the multi-valued variables x_i with Boolean indicator variables will result in polynomials with an exponential (in n) number of terms. For instance, when we encode the value of x_i as $y_{i,0} + y_{i,1} - y_{i,-1}$, and we add the additional polynomial constraints $y_{i,0} + y_{i,1} + y_{i,-1} - 1$ and $y_{i,j}^2 - y_{i,j}$, the Boolean analogue of the set P becomes $\{-\prod_{i=1}^{n}(y_{i,0} + y_{i,1} - y_{i,-1}),\ \prod_{i=1}^{n}(y_{i,0} + y_{i,1} - y_{i,-1}) - 1\}$.

Bonacina, Bonet and Levy in [3] proved that the rules in (2) are strongly sound. This is just a case analysis on the values of $p(\alpha)$, $q(\alpha)$ on an assignment α and uses the fact that $p(\alpha) + q(\alpha) \neq 0$ if and only if $(p(\alpha) + q(\alpha))^{\kappa-1} = 1$. As a consequence, the system wPC$_\kappa$ is *sound*.

▶ **Theorem 3.6** (soundness [3]). *Let $P \subseteq \mathbb{F}_\kappa[X]$ be a multi-set of polynomials. If there exists a* wPC$_\kappa$ *derivation of s copies of the polynomial* 1 *from P then* $\mathrm{cost}(P) \geqslant s$.

A converse of Theorem 3.6 (i.e., that the calculus wPC$_\kappa$ is also complete) is also true, *under the additional assumption* that the initial polynomials P are on Boolean 0/1-valued variables [3].

In Section 5 (Theorem 5.1), we complete and extend the completeness result in [3] showing a converse of Theorem 3.6 for an arbitrary κ *without any additional assumption*. Therefore allowing the use of wPC$_\kappa$ beyond multi-sets of polynomials over Boolean variables. As Example 3.5 shows, using Boolean or multi-valued variables might result in completely different complexities of the problem at hand.

4 A possible application: Max-3-Coloring

A calculus for optimization on polynomials on arbitrary variables allows using direct encoding of MaxCSPs. Moreover, the algebraic language offers the possibility to concisely encode many natural constraints used in CSP.

One notable example is the Vertex Coloring Problem (VCP), which we analyze in detail in this section. Given an undirected graph $G = (V, E)$, the *Vertex Coloring Problem* (VCP) is the problem of finding a labeling of the vertices of G such that no vertices along an edge share the same label. Deciding if a given graph is 3-colorable, i.e. colorable with a set of 3 labels, is one of the classical NP-complete problems.

Max-3-Coloring, the problem of maximizing the number of edges that connect 3-colored vertexes with distinct labels, is a more difficult problem. Max-2-Coloring is exactly MaxCUT, which is NP-complete, whereas 2-Coloring is in P.

To show that a given graph G is not say 3-colorable we can associate to the VCP on G a CNF formula which is satisfiable iff G is 3-colorable. Alternatively, we can associate the VCP on G to a set of polynomials with a common solution iff G is 3-colorable. To encode the VCP algebraically, we need to express restrictions of the form $x \neq y$. There are several ways of doing this. Using polynomials with coefficients in $\mathbb{F}_3$, it can be encoded as $x^2 + xy + y^2 - 1$. This polynomial evaluates to zero when $x \neq y$.

This immediately generalizes to expressing the *nogood* $\langle v_1, \ldots, v_\ell \rangle$ for variables $\langle x_1, \ldots, x_\ell \rangle$, i.e. the fact that at least for one index i, $x_i \neq v_i$. This can be done using the polynomial $\prod_{i=1}^{\ell}(x_i^2 + v_i\, x_i + v_i^2 - 1)$ which evaluates to zero exactly when there is an index i such that $x_i \neq v_i$.

Alternatively, for instance, the constraint $x \neq y$ for 3-coloring can be encoded using Fourier variables in $\mathbb{F}_4$ as $\{x^3 - 1, y^3 - 1, x^2 + xy + y^2 - 1\}$, where the first two polynomials

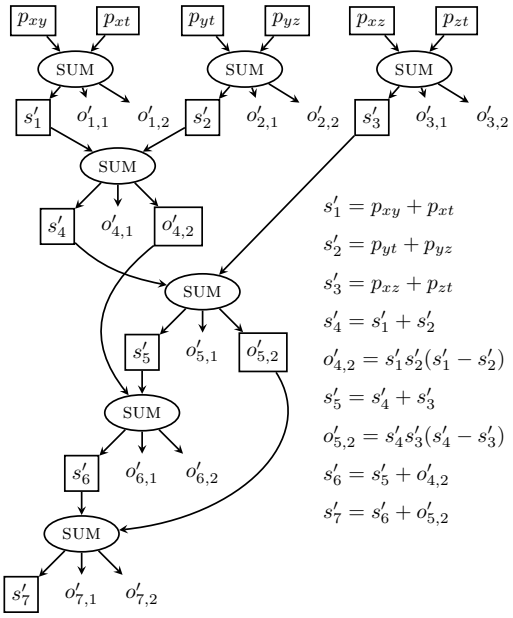

Figure 1 A proof of the non-3-colorability of K_4.

are hard constraints and the third one is a soft constraint. Both the encoding in $\mathbb{F}_3$ and the encoding in $\mathbb{F}_4$ with Fourier variables generalize easily to ℓ-coloring. In this work, for simplicity, we only consider the encoding with variables in $\mathbb{F}_3$.

▶ **Definition 4.1** ($\mathsf{VC}_3(G)$). *Given an undirected graph $G = (V, E)$, consider one variable x_v, for each $v \in V$, and the set of polynomials $\mathsf{VC}_3(G) \subset \mathbb{F}_3[\{x_v \ : \ v \in V\}]$ given by*

$$\mathsf{VC}_3(G) = \{x_v^2 + x_v x_w + x_w^2 - 1 \ : \ \{v, w\} \in E\} \ .$$

▶ **Proposition 4.2.** *The polynomials in $\mathsf{VC}_3(G)$ have a common zero iff G is 3-colorable. Moreover, $\mathrm{cost}(\mathsf{VC}_3(G))$ equals the minimum number of edges connecting two vertexes with the same color in an optimal coloring.*

Proof. It is immediate to see that $x_v^2 + x_v x_w + x_w^2 - 1 = 0$ if and only if $x_v \neq x_w$. This could be done either by analyzing all cases or, alternatively, noticing that if $x_v \neq x_w$, then

$$0 = \frac{x_v - x_w}{x_v - x_w} - 1 = \frac{x_v^3 - x_w^3}{x_v - x_w} - 1 = x_v^2 + x_v x_w + x_w^2 - 1 \ .$$

If $x_v = x_w$, then $x_v^2 + x_v x_w + x_w^2 - 1 = 3x_v^2 - 1 = -1$. ◀

In the following, as an example, we consider the 3-colorability of the complete graph $G = K_4$, *i.e.,* the graph with 4 vertexes x, y, z, t, and 6 edges connecting all pairs of vertexes. For $v, w \in \{x, y, z, t\}$, let $p_{vw} = v^2 + v\,w + w^2 - 1$, that is

$$\mathsf{VC}_3(K_4) = \{p_{xy}, p_{xz}, p_{xt}, p_{yz}, p_{yt}, p_{zt}\} \ . \tag{4}$$

In all optimal 3-colorings of K_4, one edge connects two vertexes with the same color, i.e. we must have that $\mathrm{cost}(\mathsf{VC}_3(K_4)) = 1$.

In Fig. 1 we show a wPC_3 derivation certifying that $\mathrm{cost}(\mathsf{VC}_3(K_4)) \geqslant 1$. An alternative derivation is in Fig. 2.

Every time that a polynomial is used in one inference, it is removed and cannot be used in other inferences. Therefore, every polynomial has a unique outgoing arrow.

In the wPC$_3$ derivation in Fig. 1, we obtain the polynomial s'_7 which is

$$\begin{aligned} s'_7 &= s'_1 + s'_2 + s'_3 + o'_{4,2} + o'_{5,2} \\ &= x^2y^2 - x^2yz - xy^2z - xyz^2 + y^2z^2 - x^2yt - xy^2t \\ &\quad + x^2zt - y^2zt + xz^2t - yz^2t - xyt^2 + y^2t^2 + xzt^2 \\ &\quad - yzt^2 + x^2 + xy + xz + yz + z^2 + xt + yt \\ &\quad + zt + t^2 + 1 \ . \end{aligned}$$

To complete the proof and certify that $\mathrm{cost}(\mathsf{VC}_3(K_4)) \geqslant 1$, we need to prove that s'_7 has no roots. This can be proved efficiently in wPC$_3$ following the general structure of the wPC$_3$ derivation in Proposition 5.5.

To prove that $\mathrm{cost}(\mathsf{VC}_3(K_4)) = 1$, we need to show that there exists an assignment that simultaneously evaluates to zero all the other polynomials. Obviously, this is not trivial.

An alternative wPC$_3$ derivation for the set of polynomials in (4) is in Fig. 2. In this case, we obtain the polynomial s_5, which directly proves $\mathrm{cost}(\mathsf{VC}_3(K_4)) \geqslant 1$ since $s_5 \equiv -1$. In this second proof, we use the $R_x\,(p)$ operator defined in (5) and the saturation strategy from the completeness theorem.

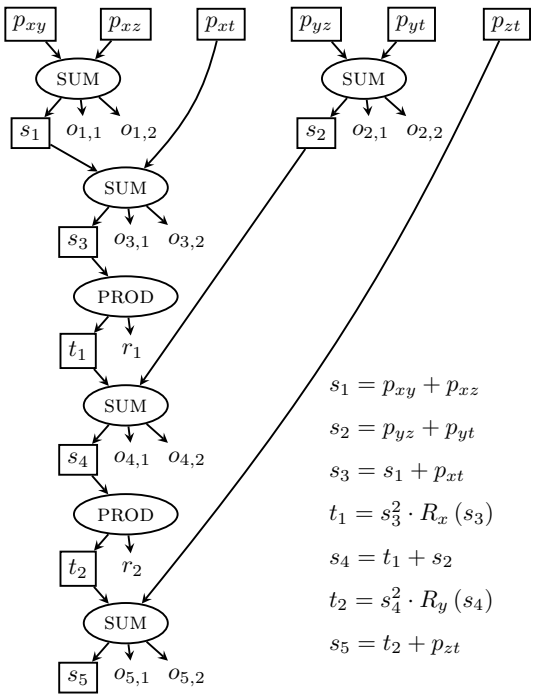

Figure 2 Alternative proof of the non-3-colorability of K_4.

In the following section, we prove a completeness result ensuring that, instead of just an unsolvable polynomial, we can get $\mathrm{cost}(\mathsf{VC}_3(K_n))$ copies of the polynomial 1 and a set of simultaneously solvable polynomials.

5 Completeness

In this section, we prove the converse of Theorem 3.6.

▶ **Theorem 5.1** (completeness). *Let $P \subset \mathbb{F}_\kappa[X]$ be a multi-set of polynomials. If $\mathrm{cost}(P) = s$, then there exists a wPC_κ derivation of a multi-set containing s copies of the polynomial 1 from P.*

The proof of Theorem 5.1 follows the structure of [3, Theorem 4.1] which was ultimately inspired by [5]. Informally, the argument goes by the following algorithmic procedure. Take an ordering on the variables $X = \{x_1, \ldots, x_n\}$, then on the polynomials *depending* on x_1 try to infer as many non-trivial polynomials without x_1 as possible using the SUM and PROD rules. This would be, intuitively, the notion of *saturation* (Definition 5.6). Then repeat the process one variable at a time. At the end of the process, we will have obtained s-many copies of the polynomial 1 together with several other polynomials. The process will have preserved the cost along the way, while the remaining polynomials are satisfiable, by the structural properties of the saturation. To make this very high-level structure of the argument formal, first, we need to formalize the notion of when a polynomial *depends* on a variable x.

▶ **Definition 5.2** (dependence). *A polynomial $p \in \mathbb{F}_\kappa[X]$* does not depend *on a variable $x \in X$ if there exists a polynomial $q \in \mathbb{F}_\kappa[X]$ not containing x such that $p \equiv q$.*

For instance, the polynomial $x^3 - x - 1 \in \mathbb{F}_3[x]$ does not depend on x, since $x^3 - x - 1 \equiv -1$ over $\mathbb{F}_3$.

The following proposition gives an equivalent characterization for when a polynomial does not depend on a variable.

For $p \in \mathbb{F}_\kappa[X]$, we use the notation $p_{x \mapsto a}$, where $x \in X$, to denote the polynomial p where each occurrence of x has been substituted with $a \in \mathbb{F}_\kappa$, and we use the notation

$$R_x(p) = \prod_{a \in \mathbb{F}_\kappa} p_{x \mapsto a} \ .$$

In particular, for $p \in \mathbb{F}_3[X]$ and $x \in X$,

$$R_x(p) = p_{x \mapsto 0} \cdot p_{x \mapsto 1} \cdot p_{x \mapsto -1} \ . \tag{5}$$

▶ **Proposition 5.3.** *Let $p \in \mathbb{F}_\kappa[X]$ be a polynomial and $x \in X$ a variable. Then the following are equivalent*

1. *p does not depend on x,*
2. *$p \equiv R_x(p)$.*

Proof. Clearly item 2 implies item 1, since p is equivalent to the polynomial $R_x(p)$ that does not contain x.

Conversely, if p does not depend on x, then there exists q not containing the variable x such that $p \equiv q$. Then, restricting by $x \mapsto a$ for all $a \in \mathbb{F}_\kappa$, we get $p_{x \mapsto a} \equiv q_{x \mapsto a} \equiv q$. Which gives $p \equiv p_{x \mapsto a}$ for each $a \in \mathbb{F}_\kappa$. Hence

$$R_x(p) = \prod_{a \in \mathbb{F}_\kappa} p_{x \mapsto a} \equiv p^\kappa \equiv p \ . \qquad \blacktriangleleft$$

The polynomials $R_x(p)$ play a central role in our argument for Theorem 5.1. In particular, it is possible to derive $R_x(p)$ from p using the rules of wPC_3.

▶ **Lemma 5.4.** *From any polynomial $p \in \mathbb{F}_\kappa[X]$, it is possible to derive $R_x(p)$ using PROD and the equivalence rule.*

Proof. The derivation is the following:

$$\dfrac{\dfrac{p}{p^{\kappa-1}\cdot R_x\left(p\right)\quad p\big(\left(p^{\kappa-2}\cdot R_x\left(p\right)\right)^{\kappa-1}-1\big)}\ \equiv}{R_x\left(p\right)\qquad p\big(\left(p^{\kappa-2}\cdot R_x\left(p\right)\right)^{\kappa-1}-1\big)}\ \text{PROD} \tag{6}$$

In the last equivalence, we used the fact that $p^{\kappa-1}\cdot R_x\left(p\right)\equiv R_x\left(p\right)$. To see this, we just need to show that for every assignment $\alpha : X\to\mathbb{F}_\kappa$,

$$p^{\kappa-1}(\alpha)\cdot R_x\left(p\right)(\alpha)\equiv R_x\left(p\right)(\alpha)\ . \tag{7}$$

Let $b=\alpha(x)$. Then, the equivalence in (7) follows from the following equalities:

$$\begin{aligned} p^{\kappa-1}(\alpha)\cdot R_x\left(p\right)(\alpha) &= p^{\kappa-1}(\alpha)\cdot\prod_{a\in\mathbb{F}_\kappa} p_{x\mapsto a}(\alpha)\\ &= p^{\kappa}(\alpha)\cdot\prod_{a\in\mathbb{F}_\kappa\setminus\{b\}} p_{x\mapsto a}(\alpha)\\ &\equiv p(\alpha)\cdot\prod_{a\in\mathbb{F}_\kappa\setminus\{b\}} p_{x\mapsto a}(\alpha)\\ &= R_x\left(p\right)(\alpha)\ . \end{aligned}$$

◀

Towards proving Theorem 5.1, first consider a generalization of Example 3.4.

▶ **Proposition 5.5.** *If $p\in\mathbb{F}_\kappa[X]$ is a polynomial that never vanishes, i.e.* $\mathrm{cost}(\{p\})=1$*, then there is a* wPC_κ *derivation of* 1 *from* $\{p\}$*.*

Proof. Let the variables in X be $x_1,\ldots,x_n$. A possible derivation is to consider $p_0=p$ and, for each $i=0,\ldots,n-1$, consider

$$p_{i+1}=R_{x_{i+1}}\left(p_i\right)\ . \tag{8}$$

By induction on i, we prove that p_i never vanishes and p_i does not depend on $x_1,\ldots,x_i$. By assumption, this holds for $i=0$. Suppose then p_i never vanishes and p_i does not depend on $x_1,\ldots,x_i$. We have two cases.

If p_i does not depend on x_{i+1}, then, by Proposition 5.3, $p_{i+1}\equiv p_i$ and the inductive hypothesis is implied.

If p_i does depend on x_{i+1}, then p_{i+1}, by construction does not depend on x_{i+1}. We need to show that p_{i+1} never vanishes. Suppose towards a contradiction it vanishes in α. Evaluating (8) in α we get:

$$0=p_{i+1}(\alpha)=R_{x_{i+1}}\left(p_i\right)(\alpha)=\prod_{a\in\mathbb{F}_\kappa} p_{i,x_{i+1}\mapsto a}(\alpha)=\prod_{a\in\mathbb{F}_\kappa} p_i(\alpha_a)$$

where α_a is the assignment α modified such that $x_{i+1}\mapsto a$. A contradiction immediately arises from the fact that p_i never vanishes, while the previous chain of equalities implies that $p_i(\alpha_a)=0$ for some $a\in\mathbb{F}_\kappa$.

The sequence of polynomials $p_0,\ldots,p_n$ gives a backbone of a wPC_κ derivation: indeed, by Lemma 5.4, each p_i is derivable from p_{i-1} and p_n is equivalent to 1. ◀

The argument in Proposition 5.5, give rise to the notion of *saturated* set of polynomials, and the extension of the previous argument to saturated sets is Lemma 5.9, the main lemma needed to prove Theorem 5.1.

▶ **Definition 5.6** (x-saturated). *A set of polynomials $S \subseteq \mathbb{F}_\kappa[X]$ is* x-saturated *if for all ℓ-tuple $p_1, \ldots, p_\ell \in S$ depending on x and for all ℓ-tuple $a_1, \ldots, a_\ell \in \mathbb{F}_\kappa$,*

$$R_x\left(\sum_{i=1}^{\ell} a_i p_i\right) \equiv 0 \ .$$

Some combinations of polynomials in the definition above are trivially zero or are redundant. For instance, $R_x(p) \equiv 0$ if and only if $R_x(-p) \equiv 0$.

The notion of x-saturated set of polynomials (Definition 5.6) is more general than what is actually needed to prove Lemma 5.9 and therefore Theorem 5.1. It is enough to check that

$$\begin{aligned} R_x(p_1) &\equiv 0 \\ R_x(\omega p_1 + p_2) &\equiv 0 \\ R_x(\omega p_1 + \omega p_2 + p_3) &\equiv 0 \\ &\vdots \\ R_x(\omega p_1 + \cdots + \omega p_{\kappa-2} + p_{\kappa-1}) &\equiv 0 \\ R_x(p_1 + \cdots + p_\kappa) &\equiv 0 \end{aligned}$$

for every $\omega \in \mathbb{F}_\kappa \setminus \{0\}$ and polynomials $p_j \in \mathbb{F}_\kappa[X]$ depending on x.

First, we show that for any set of polynomials $P \subset \mathbb{F}_\kappa[X]$ it is possible to derive in wPC_κ an x-saturated set S.

▶ **Lemma 5.7.** *For polynomials $p_1, \ldots, p_\ell \in \mathbb{F}_\kappa[X]$, coefficients $a_1, \ldots, a_\ell \in \mathbb{F}_\kappa$ and $x \in X$, there is a wPC_κ derivation from P of a multi-set containing $R_x\left(\sum_{i=1}^{\ell} a_i p_i\right)$.*

Proof. To derive in wPC_κ the polynomial $R_x\left(\sum_{i=1}^{\ell} a_i p_i\right)$ we first use the PROD and SUM rules to derive $\sum_{i=1}^{\ell} a_i p_i$ and then apply the construction in Lemma 5.4 to $\sum_{i=1}^{\ell} a_i p_i$ deriving $R_x\left(\sum_{i=1}^{\ell} a_i p_i\right)$. ◀

▶ **Lemma 5.8.** *For any multi-set of polynomials $P \subseteq \mathbb{F}_\kappa[X]$ and $x \in X$, there is a derivation from P in wPC_κ of an x-saturated multi-set of polynomials S.*

The proof of this lemma is similar to analoguos lemmas in [5, 3].

Proof. We start from P and whenever we have polynomials $p_1, \ldots, p_\ell \in P$ depending on x and coefficients $a_1, \ldots, a_\ell \in \mathbb{F}_\kappa$ such that $R_x\left(\sum_{i=1}^{\ell} a_i p_i\right) \not\equiv 0$ we apply the construction from Lemma 5.7 and get a new multi-set of polynomials P' with the property that for every α, $\mathrm{cost}(P, \alpha) = \mathrm{cost}(P', \alpha)$ but in P' we have at least one more polynomial q without x and $q \not\equiv 0$. This polynomial q is $R_x\left(\sum_{i=1}^{\ell} a_i p_i\right)$. For a set of polynomials Q, let $\sigma(Q) = \sum_\alpha \mathrm{cost}(Q_x, \alpha)$, where Q_x is the multi-set of polynomials in Q depending on x. It is always the case that $\sigma(Q) \geqslant 0$.

From the fact that $q \not\equiv 0$, that is $\sigma(\{q\}) > 0$, and the fact that wPC_κ rules preserve the cost, we obtain that $\sigma(P') < \sigma(P)$. We exhaustively apply this construction and in at most $\sigma(P)$ many steps we must reach a set of polynomials S which is x-saturated. ◀

The main property of saturated sets of polynomials is that they behave very nicely w.r.t. assignments. The following is the main technical lemma of the article.

▶ **Lemma 5.9.** *If $S \subset \mathbb{F}_\kappa[X]$ is an x-saturated multi-set of polynomials, then every assignment $\alpha : X \to \mathbb{F}_\kappa$ can be modified in x to an assignment satisfying all the polynomials in S depending on x.*

Proof. Suppose towards a contradiction, there is an assignment α and polynomials f_b for $b \in \mathbb{F}_\kappa$ in S depending on x such that $f_b(\alpha_b) \neq 0$ for every $b \in \mathbb{F}_\kappa$, where α_b is the assignment mapping x to b and every other value as in α. For shortness let $f_{b,j} = f_b(\alpha_j)$, that is, in particular $f_{b,b} \neq 0$. We just need to prove that for each $b \neq b'$, $f_{b,b'} = 0$. This will imply a contradiction in the following way: Since S is saturated,

$$R_x\left(\sum_{b\in\mathbb{F}_\kappa} f_b\right) = \prod_{j\in\mathbb{F}_\kappa} (\sum_{b\in\mathbb{F}_\kappa} f_b)_{x\mapsto j} \equiv 0 \ ,$$

and applying α we get

$$0 \overset{(\star)}{=} R_x\left(\sum_{b\in\mathbb{F}_\kappa} f_b\right)(\alpha) = \prod_{j\in\mathbb{F}_\kappa} (\sum_{b\in\mathbb{F}_\kappa} f_{b,j}) \overset{(\star\star)}{=} \prod_{j\in\mathbb{F}_\kappa} f_{j,j} \neq 0 \ ,$$

since $f_{j,j} \neq 0$ for any $j \in \mathbb{F}_\kappa$. The equality $(\star)$ holds since S is saturated and applying α, while the equality $(\star\star)$ holds if we manage to prove $f_{b,j} = 0$ for any $j \neq b$. The rest of the argument is to prove that for each $b \neq b'$, $f_{b,b'} = 0$. This will be a consequence of S being x-saturated. We consider all the polynomials of the form $\omega f_{b_1} + \omega f_{b_2} + \cdots + \omega f_{b_{t-1}} + f_{b_t}$, for $1 \leqslant t < \kappa$ and $\omega \in \mathbb{F}_\kappa \setminus \{0\}$, and proceed by induction on t.

For $t = 1$, since the saturation process stopped in S we must have that, for every $b \in \mathbb{F}_\kappa$,

$$R_x\left(f_b\right) = \prod_{j\in\mathbb{F}_\kappa} (f_b)_{x\mapsto j} \equiv 0 \ ,$$

and, since $(f_b)_{x\mapsto j}(\alpha) = f_b(\alpha_j) =: f_{b,j}$, we get

$$\prod_{j\in\mathbb{F}_\kappa} f_{b,j} = 0 \ . \tag{9}$$

In turn, this implies

$$\prod_{j\in\mathbb{F}_\kappa\setminus\{b\}} f_{b,j} = 0 \ , \tag{10}$$

since by assumption $f_{b,b} \neq 0$.

For $t = 2$, again, since S is saturated, for any b, b' distinct elements of $\mathbb{F}_\kappa$ and any $\omega \in \mathbb{F}_\kappa$, $\prod_{j\in\mathbb{F}_\kappa} (\omega f_b + f_{b'})_{x\to j} \equiv 0$.

Evaluating in α we get

$$\begin{aligned}
0 &= \prod_{j\in\mathbb{F}_\kappa} (\omega f_b + f_{b'})_{x\mapsto j}(\alpha) \\
&= \prod_{j\in\mathbb{F}_\kappa} (\omega f_b(\alpha_j) + f_{b'}(\alpha_j)) \\
&= \prod_{j\in\mathbb{F}_\kappa} (\omega f_{b,j} + f_{b',j}) \\
&= \sum_{S\subseteq\mathbb{F}_\kappa} \omega^{|S|} \prod_{j\in S} f_{b,j} \prod_{j\notin S} f_{b',j} \\
&= \sum_{\ell=0}^{\kappa} \omega^\ell \sum_{S\subseteq\binom{\mathbb{F}_\kappa}{\ell}} \prod_{j\in S} f_{b,j} \prod_{j\notin S} f_{b',j} \\
&\overset{(\star)}{=} \sum_{\ell=1}^{\kappa-1} \omega^\ell \sum_{S\subseteq\binom{\mathbb{F}_\kappa}{\ell}} \prod_{j\in S} f_{b,j} \prod_{j\notin S} f_{b',j} \ ,
\end{aligned}$$

where with $\binom{\mathbb{F}_\kappa}{\ell}$ we denoted the set of all subsets of $\mathbb{F}_\kappa$ of size ℓ. The equality in $(\star)$ is due to (9).

Let $A_\ell = \sum_{S\subseteq\binom{\mathbb{F}_\kappa}{\ell}} \prod_{j\in S} f_{b,j} \prod_{j\notin S} f_{b',j}$. By the previous equality, we get that

$$\begin{pmatrix} \omega_1 & \omega_1^2 & \cdots & \omega_1^{\kappa-1} \\ \omega_2 & \omega_2^2 & \cdots & \omega_2^{\kappa-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{\kappa-1} & \omega_{\kappa-1}^2 & \cdots & \omega_{\kappa-1}^{\kappa-1} \end{pmatrix} \cdot \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_{\kappa-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

where $\mathbb{F}_\kappa \setminus \{0\} = \{\omega_1, \ldots, \omega_{\kappa-1}\}$. The Vandermonde matrix above is invertible, therefore the unique solution is $A_\ell = 0$ for every $\ell \in \{1, \ldots, \kappa-1\}$.

For any $3 \leqslant t \leqslant \kappa - 1$, in a similar way, starting from

$$\prod_{j\in\mathbb{F}_\kappa} (\omega f_{b_1} + \omega f_{b_2} + \cdots + \omega f_{b_{t-1}} + f_{b_t})_{x\to j} \equiv 0$$

we obtain that for every $\ell \in \{1, \kappa-1\}$

$$\sum_{\substack{S_1,\ldots,S_t \\ \text{disjoint decomposition of } \mathbb{F}_\kappa \\ |S_1|+\cdots+|S_{t-1}|=\ell}} \prod_{m=1}^{t} \prod_{j\in S_m} f_{b_m,j} = 0 \ , \tag{11}$$

where $S_1, \ldots, S_t$ being a disjoint decomposition of $\mathbb{F}_\kappa$ means that $\bigcup_{i=1}^t S_i = \mathbb{F}_\kappa$ and the S_is are disjoint, but possibly empty. Since the equality (11) holds for every t and ℓ then, by induction, we also have that the equality in (11) holds for non-empty $S_1, \ldots, S_t$, that is we have

$$\sum_{\substack{S_1,\ldots,S_t \\ \text{partition of } \mathbb{F}_\kappa \\ |S_1|+\cdots+|S_{t-1}|=\ell}} \prod_{m=1}^{t} \prod_{j\in S_m} f_{b_m,j} = 0 \ , \tag{12}$$

For $t = 2$ and $\ell = 1$ this means

$$\begin{aligned}
0 &= \sum_{S \subseteq \binom{\mathbb{F}_\kappa}{1}} \prod_{j \in S} f_{b',j} \prod_{j \notin S} f_{b,j} \\
&= \sum_{r \in \mathbb{F}_\kappa} f_{b',r} \prod_{j \in \mathbb{F}_\kappa \setminus \{r\}} f_{b,j} \\
&\overset{\text{(due to (10))}}{=} \sum_{r \in \mathbb{F}_\kappa \setminus \{b\}} f_{b',r} \prod_{j \in \mathbb{F}_\kappa \setminus \{r\}} f_{b,j} \\
&= f_{b,b} \sum_{r \in \mathbb{F}_\kappa \setminus \{b\}} f_{b',r} \prod_{j \in \mathbb{F}_\kappa \setminus \{r,b\}} f_{b,j} \ .
\end{aligned}$$

Multiplying by $\prod_{j \in \mathbb{F}_\kappa \setminus \{b,b'\}} f_{b,j}$ and using (10) we get $0 = f_{b,b} f_{b',b'} \prod_{j \in \mathbb{F}_\kappa \setminus \{b,b'\}} f_{b,j}^2$, which implies

$$\prod_{j \in \mathbb{F}_\kappa \setminus \{b,b'\}} f_{b,j} = 0 \ . \tag{13}$$

For $t = 3$ the product in (13) is a single term $f_{b,j} = 0$ for $j \neq b$ arbitrary, hence concluding the argument. For an arbitrary t, we use an inductive argument with $t = \ell + 1$ and $\ell = 1, \ldots, \kappa - 2$. For simplicity we show how to go from $\ell = 1$, $t = 2$ to $\ell = 2$, $t = 3$. We have the following

$$\begin{aligned}
0 &= \sum_{\substack{S_1,S_2,S_3 \\ \text{partition of } \mathbb{F}_\kappa \\ |S_1|+|S_2|=2}} \prod_{m=1}^{3} \prod_{j \in S_m} f_{b_m,j} \\
&= \sum_{\substack{\ell_1,\ell_2 \in \mathbb{F}_\kappa \\ \ell_1 \neq \ell_2}} f_{b_1,\ell_1} f_{b_2,\ell_2} \prod_{j \in \mathbb{F}_\kappa \setminus \{\ell_1,\ell_2\}} f_{b_3,j} \\
&\overset{\text{(due to (13))}}{=} \sum_{\substack{\ell_1,\ell_2 \in \mathbb{F}_\kappa \setminus \{b_3\} \\ \ell_1 \neq \ell_2}} f_{b_1,\ell_1} f_{b_2,\ell_2} \prod_{j \in \mathbb{F}_\kappa \setminus \{\ell_1,\ell_2\}} f_{b_3,j} \\
&= f_{b_3,b_3} \sum_{\substack{\ell_1,\ell_2 \in \mathbb{F}_\kappa \setminus \{b_3\} \\ \ell_1 \neq \ell_2}} f_{b_1,\ell_1} f_{b_2,\ell_2} \prod_{j \in \mathbb{F}_\kappa \setminus \{\ell_1,\ell_2,b_3\}} f_{b_3,j} \ .
\end{aligned}$$

Multiplying by $\prod_{j \in \mathbb{F}_\kappa \setminus \{b_1,b_2,b_3\}} f_{b_3,j}$ and using (13) we get

$$0 = f_{b_1,b_1} f_{b_2,b_2} f_{b_3,b_3} \prod_{j \in \mathbb{F}_\kappa \setminus \{b_1,b_2,b_3\}} f_{b_3,j}^2 \ ,$$

which implies

$$\prod_{j \in \mathbb{F}_\kappa \setminus \{b_1,b_2,b_3\}} f_{b_3,j} = 0 \ . \tag{14}$$

Repeating this argument inductively, when $t = \kappa - 1$ and $\ell = \kappa - 2$, we get that the analogue of the product in (14) becomes a single term $f_{b_t,j} = 0$ for $j \neq b_t$. This concludes the argument for an arbitrary κ. ◀

▶ Remark 5.10. The argument for Lemma 5.9 also adapts to sets of polynomials with coefficients in $\mathbb{F}_\kappa$ *if* they are forced to take non-zero values, i.e. we consider the sets of polynomials containing $x^{\kappa-1} - 1$ for each variable x as hard constraints.

Using Lemma 5.8 and Lemma 5.9 it is then immediate to prove Theorem 5.1 using an inductive argument. This is very similar to the argument in [5, Theorem 10] and [3, Lemma 4.5].

Proof of Theorem 5.1. Choose some ordering on the set of variables $X = \{x_1, \ldots, x_n\}$. From the initial set of polynomial P using Lemma 5.8 derive using the wPC_κ rules a multi-set P_1 x_1-saturated. Let R_1 be the part of P_1 not depending on x_1 and S_1 be the part of P_1 depending on x_1. The next step is to saturate R_1 w.r.t. x_2 to get a P_2. As before let S_2 and R_2 be the sets of polynomials in P_2 depending on x_2 and not depening on x_2 (resp.). Repeat the process on R_2 saturating it w.r.t. x_3. In an inductive fashion keep doing this on all the variables one by one. Notice that that S_i depends on x_i but it does not depend on $x_1, ..., x_{i-1}$.

At the end of the process, we get s-many constant and non-zero polynomials together with the set of polynomials $S_1 \cup S_2 \cup \cdots \cup S_n$. With some trivial application of the PROD rule, we can assume that we have s-many copies of the polynomial 1.

To conclude the argument, since the cost is always preserved along the process, it is enough to show that the remaining polynomials are satisfiable. Informally, this holds because the saturated sets we produce are over fewer and fewer variables and therefore we can use Lemma 5.9 inductively on the saturated sets we produced from the last to the first one. Starting from any assignment, by Lemma 5.9 we can modify it in x_n to satisfy S_n. In general, given an assignment α_i satisfying S_i, by Lemma 5.9, we can modify it in the variable x_{i-1} obtaining an assignment α_{i-1} satisfying S_{i-1}. Since all the S_j for $j \geqslant i$ do not depend on x_{i-1} the assignment α_{i-1} continues to satisfy $S_i \cup \cdots \cup S_n$. ◀

The argument for Theorem 5.1 gives a saturation-based proof-construction strategy. Lemma 5.8 gives a possible way to obtain a saturated set, but any other heuristics to obtain a saturated set would be equally good. We exemplify the saturation strategy in Section 6.

▶ Remark 5.11 (On the completeness). To prove Theorem 5.1 we used very little of the actual form of the SUM and PROD rules. We indeed only used the fact that there are strongly sound inference rules that given as premise a polynomial p allow to infer $p\,q$ (together with other polynomials), for an arbitrary q, and that given as premises p and q allow to infer $p + q$ (together with other polynomials). In other words, Theorem 5.1 will hold for any set of strongly sound rules satisfying the properties above.

6 Another example: Flow in graphs

We exemplify the effectiveness of the saturation process on a simple constraint satisfaction problem. Given a connected graph $G = (V, E)$, an order on its vertices, and a function $b : V \to \mathbb{Z}$, find a flow for every edge such that $b(v)$ is the sum of the flows of all adjacent edges, i.e. find an assignment $f : E \to \mathbb{Z}$ such that for all $v \in V$

$$b(v) = \sum_{\substack{u \in N(v) \\ u < v}} f(\{u, v\}) - \sum_{\substack{u \in N(v) \\ u > v}} f(\{u, v\}) \ . \tag{15}$$

The problem can be solved using linear programming and it has a solution if and only if $\sum_{v \in V} b(v) = 0$. However, we are interested in possible refutations when the problem has no solution. In this case, we can pick any prime κ not dividing $\sum_{v \in V} b(v)$ and refute the set of

eq. (15) modulo κ, that is show that the set of polynomials, for $v \in V$,

$$\sum_{\substack{u \in N(v) \\ u<v}} x_{u,v} - \sum_{\substack{u \in N(v) \\ u>v}} x_{v,u} - (b(v) \mod \kappa) \tag{16}$$

has no solution in $\mathbb{F}_\kappa$. We show how the saturation process in wPC_κ, independently of the order we decide to saturate variables, always easily gives a short refutation of (16).

At the beginning choose arbitrarily a variable to saturate, let it be $x_{u,v}$. There are only two polynomials in (16) depending on $x_{u,v}$, one with positive coefficient 1, one with negative coefficient -1. It is easy to see (see Lemma 6.1 below), that to saturate w.r.t. $x_{u,v}$, the only option is to apply the SUM rule to the two (linear) polynomials containing $x_{u,v}$, and the obtained set is saturated for $x_{u,v}$. In this set, there is only one polynomial without $x_{u,v}$ which is linear. We keep saturating for all the variables one by one. It is easy to see that at any given moment the set to saturate w.r.t. the current variable x only has linear polynomials and only two of them depending on x, in one x appears with coefficient 1, in the other with coefficient -1. Again it is easy to see that the only option to saturate w.r.t. x is to apply the SUM rule to the two linear polynomials depending on x and the obtained set is saturated. Repeating this process for all variables gives the refutation. Since G is connected this will also show that the set of polynomials in (16) is minimally unsatisfiable.

▶ **Lemma 6.1.** *Let $p = \ell q$ be a polynomial in $\mathbb{F}_\kappa[X]$ with ℓ a linear polynomial depending on a variable x. Then $R_x(p) \equiv 0$.*

Proof. The polynomial p has the form $p = (cx + \ell')q$ with ℓ' linear not containing the variable x and $c \neq 0$. We have that $R_x(p) = \prod_{j \in \mathbb{F}_\kappa} (cj + \ell'|_{x \to j})q|_{x \to j}$. To show that $R_x(p) \equiv 0$ it is enough to show that for every assignment $\alpha : X \to \mathbb{F}_\kappa$, $R_x(p)(\alpha) = 0$:

$$R_x(p)(\alpha) = \prod_{j \in \mathbb{F}_\kappa} (cj + \ell'|_{x \to j}(\alpha))q|_{x \to j}(\alpha) = \prod_{j \in \mathbb{F}_\kappa} (cj + \ell'(\alpha))q|_{x \to j}(\alpha)$$

which is set to 0 by the factor corresponding to $j = -\frac{\ell'(\alpha)}{c}$. ◀

7 Conclusions

We described a calculus for MaxCSP based on polynomials and variables ranging over finite fields $\mathbb{F}_\kappa$. This provided a theoretical ground for further investigations of the efficiency of algebraic methods in the context of MaxCSPs.

The calculus is proved complete, and from this proof, we derived a *saturation*-based proof-construction strategy. We exemplified the constructions on a coloring principle, the Max-3-Coloring Problem and on a Flow principle.

References

1 Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, 40:353–373, 2011. doi:10.1613/JAIR.3152.

2 Ilario Bonacina, Maria Luisa Bonet, and Jordi Levy. Polynomial calculus for MaxSAT. In *Proceedings of the 26th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, 2023. doi:10.4230/LIPIcs.SAT.2023.5.

3 Ilario Bonacina, Maria Luisa Bonet, and Jordi Levy. Polynomial calculus for optimization. *Artificial Intelligence*, 337:104208, 2024. A preliminary version of this work appeared as [2]. doi:10.1016/j.artint.2024.104208.

4 Maria Luisa Bonet, Jordi Levy, and Felip Manyà. A complete calculus for Max-SAT. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing, (SAT)*, volume 4121 of *LNCS*, pages 240–251. Springer, 2006. doi:10.1007/11814948_24.

5 Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8-9):606–618, 2007. doi:10.1016/j.artint.2007.03.001.

6 Michael Brickenstein and Alexander Dreyer. PolyBoRi: A framework for Groebner-basis computations with Boolean polynomials. *Journal of Symbolic Computation*, 44(9):1326–1345, 2009. Effective Methods in Algebraic Geometry. doi:10.1016/j.jsc.2008.02.017.

7 B. Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *ACM SIGSAM Bulletin*, 10(3):19–29, aug 1976. doi:10.1145/1088216.1088219.

8 Sam Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *Journal of Computer and System Sciences*, 62(2):267 – 289, 2001. doi:10.1006/JCSS.2000.1726.

9 Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 174–183, 1996. doi:10.1145/237814.237860.

10 Jesús A. De Loera, Jon Lee, Peter N. Malkin, and Susan Margulies. Computing infeasibility certificates for combinatorial problems through Hilbert's Nullstellensatz. *Journal of Symbolic Computation*, 46(11):1260–1283, 2011. doi:10.1016/J.JSC.2011.08.007.

11 Jesús A De Loera, Susan Margulies, Michael Pernpeintner, Eric Riedl, David Rolnick, Gwen Spencer, Despina Stasi, and Jon Swenson. Graph-coloring ideals: Nullstellensatz certificates, Gröbner bases for chordal graphs, and hardness of Gröbner bases. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 133–140, 2015. doi:10.1145/2755996.2756639.

12 Jean Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 75–83, 2002. doi:10.1145/780506.780516.

13 Daniela Kaufmann, Paul Beame, Armin Biere, and Jakob Nordström. Adding dual variables to algebraic reasoning for gate-level multiplier verification. In *Proceedings of the 25th Design, Automation and Test in Europe Conference (DATE)*, pages 1431–1436, 2022. doi:10.23919/DATE54114.2022.9774587.

14 Daniela Kaufmann and Armin Biere. Nullstellensatz-proofs for multiplier verification. In *Proceedings of the 22nd International Workshop on Computer Algebra in Scientific Computing (CASC)*, pages 368–389, 2020. doi:10.1007/978-3-030-60026-6_21.

15 Daniela Kaufmann, Armin Biere, and Manuel Kauers. Verifying large multipliers by combining SAT and computer algebra. In *Proceedings of the 19th Conference on Formal Methods in Computer Aided Design (FMCAD)*, pages 28–36, 2019. doi:10.23919/FMCAD.2019.8894250.

16 Daniela Kaufmann, Armin Biere, and Manuel Kauers. From DRUP to PAC and back. In *Proceedigns of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 654–657, 2020. doi:10.23919/DATE48585.2020.9116276.

17 Javier Larrosa and Federico Heras. Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 193–198, 2005. URL: http://ijcai.org/Proceedings/05/Papers/0360.pdf.

18 Jesús A. De Loera, Jon Lee, Susan Margulies, and Shmuel Onn. Expressing combinatorial problems by systems of polynomial equations and Hilbert's Nullstellensatz. *Combinatorics, Probability and Computing*, 18(4):551–582, 2009. doi:10.1017/S0963548309009894.

19 Sasank Mouli. Polynomial calculus sizes over the Boolean and Fourier bases are incomparable. In *Proceedings of the 65th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 790–796, 2024. doi:10.1109/FOCS61266.2024.00055.

20 Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011. doi:10.1016/j.artint.2010.10.002.

21 Toby Walsh. SAT v CSP. In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP)*, volume 1894, pages 441–456, 2000. doi:10.1007/3-540-45349-0_32.