Artificial Intelligence 337 (2024) 104208

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint

Polynomial calculus for optimization [☆]

Ilario Bonacina [a,*], Maria Luisa Bonet [a], Jordi Levy [b]

[a] *UPC Universitat Politecnica de Catalunya, Spain*

[b] *Artificial Intelligence Research Institute (IIIA), Spanish Research Council (CSIC), Spain*

ARTICLE INFO

MSC:
03F20
68T15
03B05
03B35
68T20

Keywords:
MaxSAT
SAT
Proof systems
Polynomial calculus
Algebraic reasoning
Proof complexity

ABSTRACT

MaxSAT is the problem of finding an assignment satisfying the maximum number of clauses in a CNF formula. We consider a natural generalization of this problem to generic sets of polynomials and propose a weighted version of Polynomial Calculus to address this problem.

Weighted Polynomial Calculus is a natural generalization of the systems MaxSAT-Resolution and weighted Resolution. Unlike such systems, weighted Polynomial Calculus manipulates polynomials with coefficients in a finite field and either weights in $\mathbb{N}$ or $\mathbb{Z}$. We show the soundness and completeness of weighted Polynomial Calculus via an algorithmic procedure.

Weighted Polynomial Calculus, with weights in $\mathbb{N}$ and coefficients in $\mathbb{F}_2$, is able to prove efficiently that Tseitin formulas on a connected graph are minimally unsatisfiable. Using weights in $\mathbb{Z}$, it also proves efficiently that the Pigeonhole Principle is minimally unsatisfiable.

1. Introduction

The question of whether a set of polynomials $F = \{f_1, \dots, f_m\}$ is satisfiable—i.e. to know if there exists an assignment of the variables α s.t. $f_i(\alpha) = 0$ for every i—is at the root of algebraic geometry, and it is a natural generalization of SAT, since we can encode CNF formulas as sets of polynomials (over $\{0, 1\}$-valued variables).

The state-of-the-art practical SAT solving is dominated by CDCL SAT solvers, all of them based on the Resolution proof system [2,3]. In the last two decades, these solvers have reached remarkable efficiency in industrial SAT instances, even adding rules that deal with new variables, for instance to pre-process the given instance [4, Chapter 9]. To get further substantial improvements we think it will be necessary to broaden the current paradigm beyond Resolution. Therefore it makes sense to look at the problem from a different point of view, and using algebraic language and methods might have an impact on solving instances.

Another line of investigation is focusing on encodings to overcome the limitations of CDCL solving. For instance, [5,6] has shown that the dual-rail encoding allows translating SAT instances into MaxSAT problems. This results in translations of the Pigeonhole Principle with polynomial size proofs using MaxSAT resolution. The same applies to the translation of SAT to Max2SAT using the gadget described in [7]. Moreover, in both cases, general-purpose MaxSAT solvers are able to solve these instances in practice, even though these MaxSAT solvers are not based on MaxSAT-Resolution.

[☆] A preliminary version of this work appeared in the conference SAT 2023 [1].

[*] Corresponding author.

E-mail addresses: ilario.bonacina@upc.edu (I. Bonacina), bonet@cs.upc.edu (M.L. Bonet), levy@iiia.csic.es (J. Levy).

https://doi.org/10.1016/j.artint.2024.104208

Received 29 July 2023; Received in revised form 25 July 2024; Accepted 18 August 2024

Available online 29 August 2024

There are algebraic systems that are stronger than Resolution, and therefore it makes sense to extend those systems to solve MaxSAT problems to see if we can improve on the dual-rail and Max2SAT translations. Moreover, algebraic systems inherently allow more alternative encodings of the problems. For instance, we can use a direct encoding into polynomials, or encode first via a CNF and then translate them into polynomials. These encodings allow us to use algorithms to compute Groebner bases [8–10], and efficiency may be gained by these translations. For instance, a direct algebraic encoding, and Groebner-based techniques are useful in practice for coloring [11–13] and the verification of multiplier circuits [14–17]. The proof system capturing Groebner-based algorithms is *Polynomial Calculus* (PC) [18], which is a proof system strictly stronger than Resolution. Polynomial Calculus is degree-automatable, in the sense that bounded degree proofs can be found efficiently (in time $n^{O(d)}$, where d is the degree). This is one more reason to extend PC techniques to MaxSAT.

In this paper, we consider the generalization of MaxSAT to the context of polynomials, that is the question of what is the maximum number of polynomials of a given set we are able to simultaneously satisfy. Equivalently, the minimum number of polynomials that we cannot satisfy. In this algebraic context, there are also MaxSAT problems that have natural direct encodings as sets of polynomials, for instance, max-cut (see Example 2.3) or max-coloring. Therefore a generalization of MaxSAT to the context of polynomials and a generalization of PC to MaxSAT might be beneficial to solving these problems more efficiently.

We define an extension of PC suitable for MaxSAT, i.e. a system that not only is able to prove that a set of polynomials is unsatisfiable but to prove what is the maximum number of polynomials that can be satisfied simultaneously.

Our generalization of PC to a system suitable for MaxSAT is done in a similar way as the adaptation of Resolution to systems suitable for MaxSAT, such as MaxSAT-Resolution [19,20] and weighted Resolution [21,22]. Therefore, we think that generalizing PC to MaxSAT might be relevant to the optimization instances where PC reasoning is useful in the decision version (e.g. coloring and multiplier circuits).

As weighted Resolution is a system for MaxSAT handling *weighted* clauses, we consider *weighted* PC, a system handling weighted polynomials. We consider polynomials over *finite* fields. The intuitive reason for this is that, over a finite field $\mathbb{F}_q$ with q elements, for any non-zero element a of $\mathbb{F}_q$, $a^{q-1} = 1$. Therefore we can express a polynomial inequality $f \neq 0$ as the polynomial equality $f^{q-1} = 1$. We define weighted Polynomial Calculus for polynomials with coefficients in $\mathbb{F}_2$ in Section 4 and in Section 7 we give the definition in the general case.

We call $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ the weighted version of Polynomial Calculus that handles weighted polynomials with coefficients in $\mathbb{F}_2$ and weights in $\mathbb{N}$. Intuitively the positive weight of a clause/polynomial is the "penalty" we pay to falsify it. Weighted Resolution has been generalized to $\mathbb{Z}$-weighted Resolution, i.e. weighted resolution but with negative weights [23,21,24,25]. In a similar way, we also define $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ as $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ but where we allow negative weights in the proofs. Intuitively the meaning of a weighted clause/polynomial with a negative weight is that it is introduced in the proof as an "assumption" to be justified later, and the negative weight is to keep track of such assumptions yet to be justified.

The system $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ is strictly stronger than $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$, which in turn is strictly stronger than MaxSAT-Resolution, and moreover $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ is also strictly stronger than $\mathbb{Z}$-weighted Resolution (aka Sherali-Adams and Circular Resolution [26,21]), for details see Section 8.

The main technical contribution of this work is the definition of the systems $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}/\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$ and the proof of the completeness of $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}$, and therefore of $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$. This is proved in detail for $\mathbb{F}_2$ in Section 5 and we show how to adapt it to the general case in Section 7. The completeness is proved via a saturation process which is a natural generalization of an analogous process used in [19,20,27] to prove the completeness of MaxSAT-Resolution.

Structure of the paper Section 2 contains all the necessary preliminaries, in particular, the definition of PC and the extension of MaxSAT to polynomials. In Section 3, we introduce and discuss the structural rules that will be used in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$. In Section 4, we give the formal definition of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$. Section 5 contains the completeness of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$. In Section 6, we give an application of the saturation process to Tseitin formulas. Section 7 shows how to generalize the definition of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ from $\mathbb{F}_2$ to a generic finite field and includes a small example to illustrate the theoretical constructions in the section. Section 8 shows the relations between $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$, $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ and other proof systems for MaxSAT. Finally, in Section 9, we give some concluding remarks.

2. Preliminaries

For $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$. We usually use capital letters to denote (multi-)sets.

2.1. Propositional formulas and MaxSAT

A *clause* C is a set of literals, i.e. Boolean variables x_i or negated Boolean variables $\neg x_i$ from a given set of variables $\{x_1, \dots, x_n\}$. A CNF formula is a set of clauses, and a k-CNF is a CNF where each clause has at most k literals. An assignment $\alpha : \{x_1, \dots, x_n\} \to \{0, 1\}$ *satisfies* a clause if it maps at least a literal to 1, where $\alpha(\neg x_i) := 1 - \alpha(x_i)$. If an assignment α maps a clause C to 0 we say that α *falsifies* C. The empty clause $\perp$ is falsified by any assignment. An assignment satisfies a CNF formula if it satisfies all the clauses in it.

Let X be a generic set of variables. To define *partial weighted MaxSAT*, we distinguish between hard and soft clauses. The hard clauses need to be satisfied, while the soft ones consist of a clause and a weight (a number in $\mathbb{N}$). The weight of a soft clause is the cost associated with falsifying it. Given a set of clauses F (the *soft* clauses) and a disjoint set of clauses H (the *hard* clauses), *Weighted*

Partial MaxSAT is the problem of finding an assignment to the variables X that satisfies all the hard clauses H, and that minimizes the sum of the weights of the falsified clauses in F.

2.2. Polynomials over finite fields

Let $\mathbb{F}_q$ be a finite field with q elements (it exists whenever $q = p^k$ for some prime p and integer k). The finite field with q elements $\mathbb{F}_q$ is unique up to isomorphism and for each element of $a \in \mathbb{F}$, $a^q = a$ and $p \cdot a = \underbrace{a + \cdots + a}_{p} = 0$.

Given a finite set of variables X, with $\mathbb{F}_q[X]$ we denote the ring of multivariate polynomials with coefficients in $\mathbb{F}_q$ and variables in X.

We denote polynomials using the letters f, g, while we use Greek letters to denote assignments. An assignment is a function $\alpha : X \to \mathbb{F}_q$ and for a polynomial $f \in \mathbb{F}_q[X]$, $f(\alpha)$ is the *evaluation* of f in α: the element of $\mathbb{F}_q$ resulting from substituting each variable x in f with $\alpha(x)$ and simplifying the resulting expression using the field operations. If $f(\alpha) = 0$ we say that α *satisfies* f. Any constant polynomial c in $\mathbb{F}_q[X]$ with $c \neq 0$ is a trivially unsatisfiable polynomial (the analogue of the empty clause).

To encode CNF formulas over the variables $\{x_1, \ldots, x_n\}$ into sets of polynomials we use the encoding with *twin variables*. That is we consider polynomials over the variables $X = \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$. The intended meaning of $\bar{x}_i$ is $1 - x_i$.

For every clause $C = \{x_i : i \in I\} \cup \{\neg x_j : j \in J\}$, we associate the monomial $M(C) = \prod_{i \in I} \bar{x}_i \prod_{j \in J} x_j$ in the twin variables X. Then a set of clauses $\{C_1, \ldots, C_m\}$ is encoded as

$$\{M(C_1),\ \ldots,\ M(C_m)\} \cup \{x_i^2 - x_i,\ x_i + \bar{x}_i - 1\ : i \in [n]\}\ . \tag{1}$$

The purpose of the polynomials $x_i^2 - x_i$ and $x_i + \bar{x}_i - 1$ is to enforce the solutions of (1) to take Boolean values $0, 1$ and to enforce the variables x_i and $\bar{x}_i$ to take opposite values. Any assignment $\alpha : \{x_1, \ldots, x_n\} \to \{0, 1\}$ can be extended to an assignment $\alpha' : X \to \{0, 1\}$, where for each $i \in [n]$, $\alpha'(x_i) = \alpha(x_i)$ and $\alpha'(\bar{x}_i) = 1 - \alpha(x_i)$. Then α satisfies a CNF formula (i.e. α maps all the clauses to 1) if and only if α' satisfies the polynomial encoding of F (i.e. α' is a common solution of the polynomials).

2.3. Polynomial calculus

The algebraic proof system Polynomial Calculus was originally introduced by Clegg et al. [18]. Even though the system can be defined for any field (or even rings, see for instance [28]), in this paper we only consider finite fields.

Polynomial Calculus over $\mathbb{F}_q$ ($\mathsf{PC}_{\mathbb{F}_q}$) is a proof system that handles polynomials in $\mathbb{F}_q[X]$. A derivation in $\mathsf{PC}_{\mathbb{F}_q}$ of a polynomial f from a set of polynomials F is a sequence of polynomials $f_1, \ldots, f_m$, where $f_m = f$, and for each i either $f_i \in F$, or $f_i = g f_j$ for some $g \in \mathbb{F}_q[X]$ and $j < i$, or $f_i = f_j + f_k$ for some $j, k < i$. In other words, $\mathsf{PC}_{\mathbb{F}_q}$ uses the following two inference rules

$$\frac{f \quad g}{f + g}\ , \qquad \frac{f}{g f}\ ,$$

for all $f, g \in \mathbb{F}_q[X]$.

A *refutation* is a derivation of the polynomial 1, and the *size* of a derivation is the total number of bits needed to express it.

Sometimes, the inference rules of $\mathsf{PC}_{\mathbb{F}_q}$ are given as

$$\frac{f \quad g}{f + g}\ , \qquad \frac{f}{\alpha f} \qquad \text{and} \qquad \frac{f}{x f}$$

for all $f, g \in \mathbb{F}_q[X]$, $\alpha \in \mathbb{F}_q$, and $x \in X$. This will just give a polynomial increase in the size of the derivations (hence it is p-equivalent to our presentation of $\mathsf{PC}_{\mathbb{F}_q}$). $\mathsf{PC}_{\mathbb{F}_q}$—together with an encoding of formulas into polynomials—is a Cook-Reckhow propositional proof system [29].

With the twin encoding of CNF formulas into polynomials, it is well-known that for every q, $\mathsf{PC}_{\mathbb{F}_q}$ p-simulates Resolution and indeed the p-simulation is strict [28]. For example, Tseitin formulas (see Section 6) have polynomial size $\mathsf{PC}_{\mathbb{F}_2}$ refutations while they require exponential size in Resolution [30].

Notice that the variables $\bar{x}_i$s of the twin-variables encoding are not strictly needed for the encoding (they could be eliminated just by substituting $1 - x_i$ for each occurrence of $\bar{x}_i$), but $\mathsf{PC}_{\mathbb{F}_q}$ with this alternative encoding does not p-simulate Resolution, not even on k-CNFs [31]. With different encodings of CNF formulas, for instance, using $\{\pm 1\}$-valued variables, $\mathsf{PC}_{\mathbb{F}_2}$ becomes incomparable with Resolution [32,33].

2.4. MaxSAT on sets of polynomials

Let X be a generic set of variables. In this paper, we generalize partial weighted MaxSAT to arbitrary polynomials in $\mathbb{F}_q[X]$. The generalization of the *hard* constraints of MaxSAT is some finite set of polynomials $H \subseteq \mathbb{F}_q[X]$, while the generalization of the *soft* constraints is a multi-set of the form

$$F = \{[f_1, w_1], \ldots, [f_m, w_m]\}\ ,$$

where $f_i \in \mathbb{F}_q[X]$ and $w_i \in \mathbb{N}$. A pair $[\,f\,,\,w\,]$ where $f \in \mathbb{F}_q[X]$ and $w \in \mathbb{Z}$ is a *weighted polynomial*. The weight w informally measures the "importance" we give to satisfying the polynomial f. In this context, we are interested in assignments α that minimize the weight of the falsified soft polynomials in F, and satisfy all the polynomials in H.

Definition 2.1 *(H-compatible assignment).* Let $H \subseteq \mathbb{F}_q[X]$. An assignment $\alpha : X \to \mathbb{F}_q$ is *H-compatible* if for every $h \in H$, $h(\alpha) = 0$.

The polynomials in H could be used to enforce specific types of assignments.

Example 2.2 *(Boolean axioms).* If $H = \{x^2 - x : x \in X\}$, the H-compatible assignments are all the functions $\alpha : X \to \{0,1\}$. We refer to this H as *Boolean axioms*. If we are over $\mathbb{F}_2$ then, equivalently, H could be taken as $\emptyset$.

In the case of twin variables $X = \{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$ and the Boolean axioms $H = \{x_i^2 - x_i, x_i + \bar{x}_i - 1 : i \in [n]\}$, the H-compatible assignments are all the functions $\alpha : X \to \{0,1\}$ satisfying the additional property that for each $i \in [n]$, $\alpha(x_i) = 1 - \alpha(\bar{x}_i)$. We refer to this case as *Boolean axioms with twin variables*. Similarly as before, if we are over $\mathbb{F}_2$, then, equivalently, H could be taken as $\{x_i + \bar{x}_i - 1 : i \in [n]\}$.

Now, for each assignment $\alpha : X \to \mathbb{F}_q$, we measure how close it is to satisfying all the polynomials in F, and we do this by defining its *cost* as the sum of the weights of the polynomials in F not satisfied by α. Therefore, the cost of the assignment α on F is

$$\operatorname{cost}(\alpha, F) = \sum_{i \in [m]} w_i \chi_i(\alpha)\ , \tag{2}$$

where $\chi_i(\alpha)$ is 1 if $f_i(\alpha) \neq 0$, and 0 otherwise. Finally, to solve a partial weighted MaxSAT problem, we want the minimum value of $\operatorname{cost}(\alpha, F)$ for any H-compatible assignment α, i.e.

$$\operatorname{cost}_H(F) = \min_{\alpha\ H\text{-compatible}} \operatorname{cost}(\alpha, F)\ . \tag{3}$$

If $H = \emptyset$, we denote $\operatorname{cost}_H(F)$ simply as $\operatorname{cost}(F)$. Of course, if F is satisfiable using a H-compatible assignment, then $\operatorname{cost}_H(F) = 0$.

Notice that, the polynomials in H and the weighted polynomials in F could come from the translation of a partial weighted MaxSAT instance. However, we cannot assume this is always the case. Sometimes a direct encoding with polynomials not coming from CNF formulas is more natural. For instance, this is the case of max-cut.

Example 2.3 *(max-cut).* Given a graph $G = (V, E)$ consider $X = \{x_v : v \in V\}$ and let $F = \{[\,x_{v_1} + x_{v_2} + 1\,,\,1\,] : (v_1, v_2) \in E\} \subseteq \mathbb{F}_2[X]$. Finding $\operatorname{cost}(F)$ is equivalent to finding a max-cut in G. This codification could be easily generalized to weighted max-cut.

3. Structural inference rules for MaxSAT calculi

In this paper, we construct calculi for MaxSAT on sets of polynomials. That is we are given as input a finite set of hard polynomials $H \subseteq \mathbb{F}_q[X]$ and a multi-set of weighted polynomials F, that is F consists of pairs $[\,f\,,\,w\,]$ with $f \in \mathbb{F}_q[X]$ and $w \in \mathbb{N}$. The calculi we consider, given F and H construct a sequence of multi-sets of weighted polynomials $L_0, \dots, L_\ell$ via the application of inference rules used as *substitution* rules, that is rules that when applied to some L_i *replace* the premises with the conclusions to get to L_{i+1}. The goal for a MaxSAT calculus is then to start from $L_0 = F$ and to get to a multiset L_ℓ containing $[\,1\,,\,c\,]$ using the inference rules proper of the calculus to certify that $\operatorname{cost}_H(F) \geq c$.

Definition 3.1 *(strong soundness).* A substitution rule is *strongly sound* if, for every assignment α, the cost of the set of premises on α equals the cost of the conclusions on α.

We give the notion of strong soundness to avoid confusion with the usual notion of sound inference rules. In this article we only consider substitution rules strongly sound.

In this section we discuss the basic structural substitution rules for our calculi for MaxSAT: the *fold-unfold* equivalence and the *H-equivalence*. In the next section we define the remaining rules.

Definition 3.2 *(fold-unfold equivalence, $\approx$).* Let F, G be two multi-sets of weighted polynomials, we say that F and G are *fold-unfold equivalent* ($F \approx G$) if there is a sequence of multi-sets of weighted polynomials starting with F and ending with G where from one multi-set to the next, one of the following substitution rules is applied

$$\frac{[\,f\,,\,u\,] \quad [\,f\,,\,w\,]}{[\,f\,,\,u+w\,]}\ (\textsc{fold}) \qquad\qquad \frac{[\,f\,,\,u+w\,]}{[\,f\,,\,u\,] \quad [\,f\,,\,w\,]}\ (\textsc{unfold})$$

$$\frac{[\,f\,,\,0\,]}{}\ (0\text{-}\textsc{fold}) \qquad\qquad \frac{}{[\,f\,,\,0\,]}\ (0\text{-}\textsc{unfold})$$

where $f \in \mathbb{F}_q[X]$, and $w, u \in \mathbb{Z}$.

Notice that, unlike the initial weighted polynomials in F which have weights in $\mathbb{N}$, in a fold-unfold equivalence we also allow weights in $\mathbb{Z}$. Negative weights can only be created using the UNFOLD rule. The fold-unfold equivalence is analogous to the fold-unfold equivalence used in [21] in the context of weighted clauses and weighted Resolution.

Example 3.3. For any polynomial f, $\{[f, 0]\} \approx \{[f, 1], [f, -1]\} \approx \emptyset$ and $\{[f, 2]\} \approx \{[f, 1], [f, 1]\}$.

It is immediate to see that the fold-unfold rules are strongly sound.

The second type of basic substitution rule is the H-equivalence.

Definition 3.4 *(H-equivalence, $\equiv_H$).* Given $f, g \in \mathbb{F}_q[X]$ and a finite set $H \subseteq \mathbb{F}_q[X]$, we say that f and g are H-equivalent ($f \equiv_{\mathbb{F}_q, H} g$) if for every H-compatible assignment $\alpha : X \to \mathbb{F}_q$, $f(\alpha) = g(\alpha)$. When the field $\mathbb{F}_q$ and H are clear from the context we write simply $\equiv$.

This gives the following substitution rule

$$\frac{[f, w]}{[g, w]}\ (\equiv_H)$$

where $f, g \in \mathbb{F}_q[X]$ and $f \equiv_H g$.

The motivation behind this rule and the notion of H-equivalence is that in the MaxSAT calculi we are always interested in satisfying the hard polynomials in H, therefore two distinct polynomials that always evaluate the same under H-compatible assignments are interchangeable. By definition, if $f \equiv_H g$ then the cost is preserved on every H-compatible α, hence the rule is strongly sound on H-compatible αs.

Example 3.5. For every $H \subseteq \mathbb{F}_q[X]$ and every $f \in \mathbb{F}_q[X]$, $f^q \equiv_H f$. This is because for every $a \in \mathbb{F}_q$, $a^q = a$, and therefore on any assignment the polynomials f^q and f give the same value.

Example 3.6. For $H = \{x + \bar{x} + 1, y + \bar{y} + 1\} \subseteq \mathbb{F}_2[x, y, \bar{x}, \bar{y}]$, we have

$$xy + \bar{x}\bar{y} + x + y \equiv_H xy + (1 + x)(1 + y) + x + y = 1\ .$$

Depending on H, to efficiently check whether $f \equiv_H g$ might be computationally expensive. This is not the case for the hard constraints H we consider in this paper, the Boolean axioms and the Boolean axioms with twin variables. We conclude this section showing how to check efficiently the H-equivalence in these two cases.

If $H = \{x^2 - x : x \in X\} \subseteq \mathbb{F}_q[X]$, the only H-compatible assignments are of the form $\alpha : X \to \{0, 1\}$, that is they are Boolean assignments. Notice that, if $q = 2$, the Boolean assignments are the only type of assignments possible and this is the same as using $H = \emptyset$. In the case of Boolean assignments, to check efficiently whether $f \equiv_H g$ we can just compute the multilinearization of both f and g and check if they are equal. The multilinearization of a polynomial f is the unique multilinear polynomial $\mathrm{mul}(f)$ obtained from f dropping all the exponents down to 1 and then simplifying using the simplification rules of the field.

Example 3.7. $f = x^3y + xy^2 + z$ and $g = z$ as polynomials over $\mathbb{F}_2[X]$ are H-equivalent, because $\mathrm{mul}(f) = \mathrm{mul}(g) = z$.

The other case we consider for the hard polynomials H are the Boolean axioms with twin variables. That is we consider polynomials over $\mathbb{F}_q[X \cup \{\bar{x} : x \in X\}]$ and $H = \{x^2 - x, x + \bar{x} - 1 : x \in X\} \subseteq \mathbb{F}_q[X \cup \{\bar{x} : x \in X\}]$. These are the type of variables used to encode CNF formulas and the polynomials in H are the part of the encoding used to ensure the variables to be Boolean and the semantic meaning of $\bar{x}$ to be $1 - x$. In this case too it is possible to check efficiently whether $f \equiv_H g$ by the algorithm in [34, section 4.3 and Theorem 4.4].

4. Polynomial calculus for MaxSAT (special case $\mathbb{F}_2$)

We first define Polynomial Calculus for MaxSAT in the special case of polynomials with coefficients in $\mathbb{F}_2$ (the general case is in Section 7).

The initial instance consists of a multi-sets of weighted polynomials, i.e. pairs $[f, w]$ with $f \in \mathbb{F}_2[X]$ and $w \in \mathbb{N}$, and a finite set of hard polynomials $H \subseteq \mathbb{F}_2[X]$. We define $\mathbb{Z}$*-weighted Polynomial Calculus* ($\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$), an inference system that handles weighted polynomials with weights in $\mathbb{Z}$, and $\mathbb{N}$*-weighted Polynomial Calculus* ($\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$), an inference system that handles weighted polynomials with weights in $\mathbb{N}$. Both systems use the same set of inference rules. The formal definition of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}/\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ is Definition 4.1, but we discuss first the inference rules of the system. They are two types: the structural rules from the previous section (the FOLD, UNFOLD and the H-EQUIVALENCE), and SUM and PROD:

$$\frac{[f, w]}{[fg, w] \quad [f(g+1), w]}\ (\text{PROD}) \qquad \frac{[f, w] \quad [g, w]}{[f+g, w] \quad [fg, 2w]}\ (\text{SUM}) \tag{4}$$

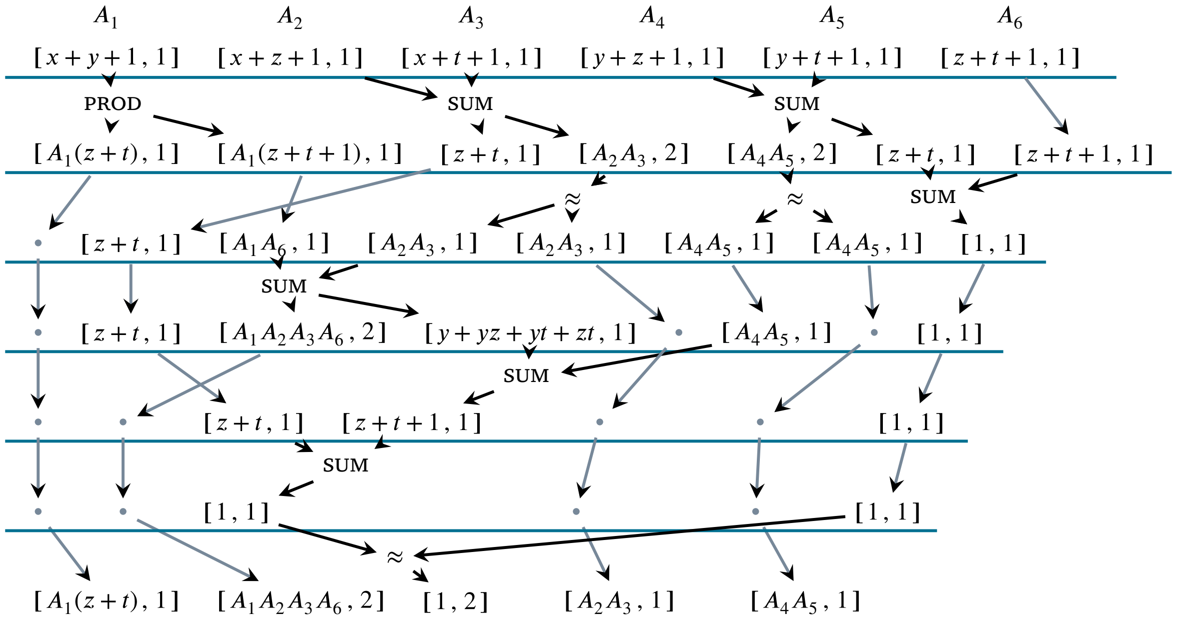
A_1 A_2 A_3 A_4 A_5 A_6
$[x+y+1,1]$ $[x+z+1,1]$ $[x+t+1,1]$ $[y+z+1,1]$ $[y+t+1,1]$ $[z+t+1,1]$
PROD SUM SUM
$[A_1(z+t),1]$ $[A_1(z+t+1),1]$ $[z+t,1]$ $[A_2A_3,2]$ $[A_4A_5,2]$ $[z+t,1]$ $[z+t+1,1]$
$\approx$ $\approx$ SUM
• $[z+t,1]$ $[A_1A_6,1]$ $[A_2A_3,1]$ $[A_2A_3,1]$ $[A_4A_5,1]$ $[A_4A_5,1]$ $[1,1]$
SUM
• $[z+t,1]$ $[A_1A_2A_3A_6,2]$ $[y+yz+yt+zt,1]$ • $[A_4A_5,1]$ • $[1,1]$
SUM
• • $[z+t,1]$ $[z+t+1,1]$ • • $[1,1]$
SUM
• • $[1,1]$ • • $[1,1]$
$\approx$
$[A_1(z+t),1]$ $[A_1A_2A_3A_6,2]$ $[1,2]$ $[A_2A_3,1]$ $[A_4A_5,1]$

Fig. 1. A $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ derivation of $[1,2]$ from the axioms of max-cut on a clique of 4 vertices: $[x_1+x_2+1,1]$, $[x_1+x_3+1,1]$, $[x_1+x_4+1,1]$, $[x_2+x_3+1,1]$, $[x_2+x_4+1,1]$, $[x_3+x_4+1,1]$. To improve the readability of the derivation we draw in grey the weighted polynomials that are just copied from one multiset to the next. The •s are placeholders for the corresponding weighted polynomials.

for all $f, g \in \mathbb{F}_2[X]$ and $w \in \mathbb{Z}$.[1]

Notice that the previous rules are strongly sound. For the PROD rule, for any assignment α, if $f(\alpha)=0$ then both the conclusions are 0, but if $f(\alpha)=1$, then exactly one of the conclusions is 1. For the SUM rule, the argument by cases is analogous. The case that justifies the weight of $2w$ for the polynomial fg in the conclusion is when $f(\alpha)=1$ and $g(\alpha)=1$. In this case $f(\alpha)+g(\alpha)=0$ and $f(\alpha)g(\alpha)=1$, hence the weight of the conclusion fg should equal the sum of the weights of both premises, which is two.

Formally the definition of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ are the following.

Definition 4.1 *($\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$).* Given a multi-set of weighted polynomials $F=\{[f_1,w_1],\dots,[f_m,w_m]\}$ with $f_i\in\mathbb{F}_2[X]$ and a finite set of hard constraints $H\subseteq\mathbb{F}_2[X]$, a $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ *derivation* of a weighted polynomial $[f,w]$ from F and H is a sequence of multi-sets $L_0,\dots,L_\ell$ s.t.

1. $L_0=F$,
2. $[f,w]\in L_\ell$ and all the other weighted polynomials $[f',w']\in L_\ell$ have $w'\in\mathbb{N}$, and
3. for each $i>0$ either $L_i\approx L_{i-1}$ or L_i is the result of an application of the PROD/SUM/$\equiv_H$ rule as a substitution rule on L_{i-1}.

The *size* of a $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ derivation $L_0,\dots,L_\ell$ is the total number of occurrences of symbols in $L_0,\dots,L_\ell$.

Definition 4.2 *($\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$).* The system $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ is the restriction of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ where all the weights appearing in the derivation are natural numbers.

To clarify the definition, we give an example of derivation in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$.

Example 4.3 *(Example 2.3 cont.).* In Fig. 1 we show a $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$-derivation of $[1,2]$ from the set of polynomials we saw in Example 2.3 in the case of G being the clique on 4 vertices. That is the weighted polynomials $[x+y+1,1]$, $[x+z+1,1]$, $[x+t+1,1]$, $[y+z+1,1]$, $[y+t+1,1]$, $[z+t+1,1]$. In this derivation, the polynomials that are just copied from one multiset to the next are substituted with a •. From one multiset to the next we applied multiple rules (and the fold-unfold equivalence) in parallel. The horizontal lines are just a visual help to visualize the multisets. Notice that we have H-equivalences (for $H=\emptyset$) applied implicitly. For instance, some SUM only have one consequence since the other is equivalent to 0. This example shows that to obtain $[1,2]$ it is important to use both consequences of a SUM.

We prove now the *soundness* of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$.

[1] In the preliminary version of this work [1] the PROD rule was called SPLIT in analogy with the SPLIT rule of weighted Resolution.

Theorem 4.4 *(soundness). Given $F = \{[f_1, w_1], \dots, [f_m, w_m]\}$ where $f_i \in \mathbb{F}_2[X]$ and a set of polynomials $H \subseteq \mathbb{F}_2[X]$, if there is a $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ derivation of $[1, w]$ from F (and H as hard constraints), then $\mathrm{cost}_H(F) \geq w$.*

Proof. Let $L_0, L_1, L_2, \dots, L_s$ be a $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ derivation of $[1, w]$, i.e. L_s contains $[1, w]$, $L_0 = F$ and each L_{i+1} is obtained from L_i applying the PROD, the SUM substitution rules, the fold-unfold equivalence or the H-equivalence. We have that $\mathrm{cost}_H(L_s) \geq w$ since $[1, w] \in L_s$ and all the other weighted polynomials in L_s have non-negative weights. Hence, to prove the statement is enough to show that for each i, $\mathrm{cost}_H(L_{i+1}) = \mathrm{cost}_H(L_i)$. We prove something slightly stronger, that for each H-consistent $\alpha : X \to \{0, 1\}$, $\mathrm{cost}(\alpha, L_{i+1}) = \mathrm{cost}(\alpha, L_i)$. This follows immediately from the comments we already made on the soundness of the various substitution rules. □

We conclude this section with a lemma on the PROD and SUM rules in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$.

Lemma 4.5. *Using weights in $\mathbb{Z}$, and the structural substitution rules, the SUM rule can simulate the PROD rule and vice versa.*

Proof. To simulate the PROD rule using the SUM rule using weights in $\mathbb{Z}$ we can do the following:

$$\begin{array}{cl}
[f, w] & \\ \cline{1-1}
[f, w]\quad [fg, w]\quad [fg, -w]\quad [f(g+1), w]\quad [f(g+1), -w] & \approx \\ \cline{1-1}
[f, w]\quad [fg, w]\quad [f(g+1), w]\quad [fg + f(g+1), -w]\quad [f^2 g(g+1), -2w] & \text{SUM} \\ \cline{1-1}
[fg, w]\quad [f(g+1), w] & \approx \,\&\, \equiv
\end{array}$$

Some steps are merged for improved clarity. In a similar way, we can also simulate the SUM using the PROD rule using weights in $\mathbb{Z}$:

$$\begin{array}{cl}
[f, w]\quad [g, w] & \\ \cline{1-1}
[f, w]\quad [g, w]\quad [f+g, w]\quad [f+g, -w] & \approx \\ \cline{1-1}
[f, w]\quad [g, w]\quad [f+g, w]\quad [(f+g)f, -w]\quad [(f+g)(f+1), -w] & \text{PROD} \\ \cline{1-1}
[f, w]\quad [g, w]\quad [f+g, w]\quad [f(f+g), -w]\quad [g(f+1), -w] & \equiv \\ \cline{1-1}
[f, w]\quad [gf, w]\quad [g(f+1), w]\quad [f+g, w]\quad [f(f+g), -w]\quad [g(f+1), -w] & \text{PROD} \\ \cline{1-1}
[f, w]\quad [gf, w]\quad [f+g, w]\quad [f(f+g), -w] & \approx \\ \cline{1-1}
[f(f+g), w]\quad [f(f+g+1), w]\quad [gf, w]\quad [f+g, w]\quad [f(f+g), -w] & \text{PROD} \\ \cline{1-1}
[f(f+g+1), w]\quad [gf, w]\quad [f+g, w] & \approx \\ \cline{1-1}
[fg, 2w]\quad [f+g, w] \quad \square & \equiv
\end{array}$$

5. Completeness

We show the completeness of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$, that is the converse of Theorem 4.4. In principle we consider three cases $H = \emptyset$, or H the Boolean axioms, i.e. $H = \{x^2 - x : x \in X\}$, or the case of twin variables $\{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$, that is $H = \{x^2 - 2, x_i + \bar{x}_i - 1 : i \in [n]\}$. Since we are working with polynomials over $\mathbb{F}_2$, the case of Boolean axioms is the same as $H = \emptyset$, and the case of twin variables is the same as $H = \{x_i + \bar{x}_i - 1 : i \in [n]\}$. We show the completeness for both sets H.

Theorem 5.1 *(completeness for Boolean axioms). Given F a set of weighted polynomials over $\mathbb{F}_2[X]$, there is a $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ derivation of $[1, \mathrm{cost}_H(F)]$ from F and the set of Boolean axioms as hard constraints H.*

Before we prove Theorem 5.1, we define some concepts and prove some relevant lemmas. Our proof generalizes the *saturation* process from [20] and gives an algorithm to find $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$-derivations of $[1, \mathrm{cost}_H(F)]$. We give an example of the construction in Section 6. Clearly the completeness for $H = \emptyset$ implies the completeness for $H = \{x_i + \bar{x}_i - 1 : i \in [n]\}$. For instance, just removing the twin variables, that is substituting each variable $\bar{x}_i$ with $1 - x_i$. This might result in exponentially larger derivations, therefore we show a saturation process that adapts naturally to twin variables without the need to remove them.

The intuition of the proof is the following. First we pick a variable x and we perform PROD and SUM inferences as long as we obtain new polynomials without the variable x. We show this process is finite and we will call it saturation. This way we create two multisets of weighted polynomials. One with polynomials that depend on a variable x, and another with polynomials that do not depend on x. We will then pick another variable to saturate multiset of weighted polynomials that do not depend on the first variable x. We will do the same with the rest of the variables always using the set of weighted polynomials that do not contain the variables we already saturated by. In the last saturation step we get a weighted polynomial $[1, w]$ and a satisfiable set of weighted polynomials. This is due to the fact that the saturation procedure maintains a good property: given a multiset of clauses saturated w.r.t. a variable x, if there exists an assignment satisfying all the polynomials not depending on x, then it can be extended (by assigning x) to satisfy all the polynomials. The weight w will correspond to the cost of the initial polynomials.

For the saturation process we need to formalize the notion of when a polynomial f depends or not on a variable x. Clearly, $f_{x\mapsto 0}$ and $f_{x\mapsto 1}$ do not contain x and therefore do not depend on it, but, f could contain the variable x but not depend on it by being equivalent to a polynomial not containing x.

Definition 5.2 *(dependence).* Let $H \subseteq \mathbb{F}_2[X]$ be a finite set. We say that a polynomial f does not *depend* on a variable x w.r.t. H if there exist a polynomial g not containing x (and also $\bar{x}$ in the case of twin variables) such that $f \equiv_H g$.

Example 5.3. The polynomial $f = xy + \bar{x}\bar{y} + x + y$ is equivalent to the polynomial 1 modulo $H = \{x + \bar{x} - 1, y + \bar{y} - 1\}$, so although it seems to "depend" on x and y, indeed it does not depend on them w.r.t. H.

Notice that in the definition of dependence and the next characterization, the set of polynomials H is completely arbitrary.

Proposition 5.4. *Let f be a polynomial in $\mathbb{F}_2[X]$, x a variable and $H \subseteq \mathbb{F}_2[X]$ be a finite set. The following are equivalent*

1. *f does not depend on x w.r.t. H;*
2. *$f \equiv_H f_{x\mapsto 0} \equiv_H f_{x\mapsto 1}$;*
3. *$f \equiv_H f_{x\mapsto 0} f_{x\mapsto 1}$.*

Proof. Since H is fixed thorough this proof we omit it, but implicitly every $\equiv$ is with respect to the fixed H. We prove that item 1 implies item 2, item 2 implies item 3, and item 3 implies item 1.

Item 1 implies item 2: If f does not depend on x, then by definition there exist g not containing x such that $f \equiv g$. Restricting $f \equiv g$ by $x = 0$ we get $f_{x\mapsto 0} \equiv g_{x\mapsto 0}$, and since g does not contain x, $g \equiv g_{x\mapsto 0}$. So $f \equiv f_{x\mapsto 0}$. Restricting by $x = 1$ we get $f \equiv f_{x\mapsto 1}$.

Item 2 implies item 3: If $f \equiv f_{x\mapsto 0} \equiv f_{x\mapsto 1}$, then $f_{x\mapsto 0} f_{x\mapsto 1} \equiv f^2 \equiv f$.

Item 3 implies item 1: The polynomial $f_{x\mapsto 0} f_{x\mapsto 1}$ does not contain x, therefore item 3 implies that f does not depend on x. □

Notice that by Proposition 5.4, to check whether a polynomial f depends or not on a variable x w.r.t. H it is enough to check whether $f \equiv_H f_{x\mapsto 0} f_{x\mapsto 1}$. Section 3 discusses effective ways to check this in the case $H = \emptyset$ and $H = \{x + \bar{x} - 1 : x \in X\}$.

Once we have the notion of a polynomial depending on x, the main concept used to show the completeness of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ is the notion of set of polynomials *fixable* w.r.t. a variable. This notion is related to the notion of saturation from [19] but is potentially more general.

Definition 5.5 *(x-fixable set).* Let $H \subseteq \mathbb{F}_2[X]$ be a finite set, $x \in X$ and S a set of weighted polynomials. The set S is *x-fixable* if every H-compatible assignment $\alpha : X \to \mathbb{F}_2$ can be modified in x to a H-compatible assignment satisfying all weighted polynomials in S that depend on x.

Notice that, if S is x-fixable then the subset of polynomials in S depending on x is satisfiable, but the converse is not true.

Example 5.6. $\{[x + y, 1], [x + z, 1]\}$ is clearly satisfiable but it is not x-fixable since we cannot extend the assignment $y = 0, z = 1$ to satisfy both polynomials.

Next, we give a procedure to construct a x-fixable set of polynomials from a given one using the rules of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ (Lemma 5.11). We focus on H being the Boolean axioms.

Informally, the procedure to obtain an x-fixable set consists of applying the PROD rule to a polynomial or the SUM of two polynomials, as long as the application of these rules generates polynomials that do not contain the variable x, and are not equivalent to 0. The procedure is applied as long as it is possible, and we will see that it finishes in a finite number of steps and when it terminates the generated set must be x-fixable.

There are several ways to accomplish this. We use the PROD rule only in the following special form

$$\frac{[f, w]}{[f f_{x\mapsto 0} f_{x\mapsto 1}, w] \quad [f(f_{x\mapsto 0} f_{x\mapsto 1} + 1), w]}, \tag{5}$$

where x is some variable and $f_{x\mapsto 0}$ is the polynomial resulting from the restriction of f mapping x to 0, and analogously for $f_{x\mapsto 1}$. In the case of twin variables, for $f_{x\mapsto 0}$ we also map $\bar{x}$ to 1, to be consistent with the Boolean axioms, and analogously for $f_{x\mapsto 1}$.

Lemma 5.7. *For $f \in \mathbb{F}_2[X]$ and every finite $H \subseteq \mathbb{F}_2[X]$, $f f_{x\mapsto 0} f_{x\mapsto 1} \equiv_H f_{x\mapsto 0} f_{x\mapsto 1}$.*

Proof. Recall that for polynomials $f, g \in \mathbb{F}_2[X]$, we defined $f \equiv g$ if for every H-compatible assignment α, $f(\alpha) = g(\alpha)$. Recall also that for every value $a \in \mathbb{F}_2$, $a^2 = a$. Setting $x = 0$ we get $f_{x\mapsto 0}f_{x\mapsto 0}f_{x\mapsto 1} \equiv (f_{x\mapsto 0})^2 f_{x\mapsto 1} \equiv f_{x\mapsto 0}f_{x\mapsto 1}$. Setting $x = 1$ we get $f_{x\mapsto 1}f_{x\mapsto 0}f_{x\mapsto 1} \equiv (f_{x\mapsto 1})^2 f_{x\mapsto 0} \equiv f_{x\mapsto 0}f_{x\mapsto 1}$. Therefore $f f_{x\mapsto 0}f_{x\mapsto 1} \equiv f_{x\mapsto 0}f_{x\mapsto 1}$. □

Therefore, by Lemma 5.7, restricting ourselves to PROD of type (5) is the same as restricting ourselves to substitutions of the form

$$\frac{[\,f\,,\,w\,]}{[\,f_{x\mapsto 0}f_{x\mapsto 1}\,,\,w\,],[\,f+f_{x\mapsto 0}f_{x\mapsto 1}\,,\,w\,]}\,. \tag{6}$$

If f depends on x and $f_{x\mapsto 0}f_{x\mapsto 1} \not\equiv 0$, performing the substitution rules from above we obtain a polynomial that does not depend on x, and it is not equivalent to f or to 0. This property will be crucial in proving the completeness and in giving a finite procedure to obtain $[\,1\,,\,\mathrm{cost}_H(F)\,]$.

For the SUM we do something similar. We SUM and then apply the special form of PROD in (6):

$$\begin{array}{c l}
[\,f\,,\,w\,]\quad[\,g\,,\,w\,] & \\ \hline
[\,f+g\,,\,w\,]\quad[\,fg\,,\,2w\,] & \textsc{sum} \\ \hline
[\,(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1}\,,\,w\,]\quad[\,(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1}+f+g\,,\,w\,]\quad[\,fg\,,\,2w\,] & \text{by (6)}
\end{array}$$

If f and g depend on x and $(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1} \not\equiv 0$, performing the substitution above we obtain a polynomial that does not depend on x, and it is not equivalent to 0.

As we mentioned, there are alternative ways to introduce new polynomials not depending on x. Our choice has two noticeable properties:

- it only uses a special form of the PROD rule, the one in (6), and
- it keeps the number and the degree of the polynomials introduced at each step lower than other alternative choices.

For instance, an alternative—perhaps more intuitive but less efficient—way could have been the following. Given multilinear polynomials $f = xf_1 + f_0$ and $g = xg_1 + g_0$ depending on x, instead of SUM and then apply the substitution inference (6), we could have done:

$$\begin{array}{c l}
[\,xf_1+f_0\,,\,w\,]\quad[\,xg_1+g_0\,,\,w\,] & \\ \hline
[\,(xf_1+f_0)g_1\,,\,w\,]\quad[\,(xf_1+f_0)(g_1+1)\,,\,w\,]\quad[\,xg_1+g_0\,,\,w\,] & \textsc{prod} \\ \hline
[\,(xf_1+f_0)g_1\,,\,w\,]\quad[\,(xf_1+f_0)(g_1+1)\,,\,w\,]\quad[\,(xg_1+g_0)f_1\,,\,w\,]\quad[\,(xg_1+g_0)(f_1+1)\,,\,w\,] & \textsc{prod} \\ \hline
[\,f_0g_1+f_1g_0\,,\,w\,]\quad[\,(xf_1+f_0)(g_1+1)\,,\,w\,]\quad[\,(xg_1+g_0)(f_1+1)\,,\,w\,]\quad[\,fgf_1g_1\,,\,2w\,] & \textsc{sum}
\end{array}$$

This way, we could also obtain a polynomial without x, the polynomial $f_0g_1 + f_1g_0$. This comes at the cost of introducing a larger number of polynomials and of higher degree compared to the construction we use for the SUM of two polynomials.

Now we are ready to define when a set of polynomials S is *x-saturated*. Informally this means that we are not able to perform substitution inferences of the form (6) or the type we use for the SUM to get new polynomials without x.

Definition 5.8 *(x-saturated set).* Let S a set of polynomials in $\mathbb{F}_2[X]$, $x \in X$ and $H = \emptyset$ or H the Boolean axioms with twin variables. The set S is *x-saturated w.r.t. H* if

- for all $f \in S$ depending on x, $f_{x\mapsto 0}f_{x\mapsto 1} \equiv_H 0$, and
- for all $f, g \in S$ depending on x, $(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1} \equiv_H 0$.

Notice that this notion can be seen as a generalization of the saturation for MaxSAT-Resolution [20] where a set of clauses is saturated w.r.t. x if all the cuts involving x are substituted by clauses without x which are tautologies.

Example 5.9. The set of clauses $S = \{y, x \vee y, \neg x \vee \neg y\}$ is saturated for x. There is only one possible cut on x which will produce the tautology $y \vee \neg y$. This example, translated to polynomials with twin variables, is the set $S' = \{\bar{y}, xy, \bar{x}\bar{y}\}$, which is x-saturated since $(xy)_{x\mapsto 0}(xy)_{x\mapsto 1} = 0$, $(\bar{x}\bar{y})_{x\mapsto 0}(\bar{x}\bar{y})_{x\mapsto 1} = 0$ and $(xy + \bar{x}\bar{y})_{x\mapsto 0}(xy + \bar{x}\bar{y})_{x\mapsto 1} = y\bar{y} \equiv_H 0$.

The saturation for MaxSAT-Resolution corresponds to the special case of Definition 5.8 where all polynomials in S are monomials with twin variables.

Lemma 5.10. *When $H = \emptyset$ or H are Boolean axioms with the twin variables, if S is x-saturated w.r.t. H, then S is x-fixable w.r.t. H.*

Proof. Suppose, towards a contradiction, that S is not x-fixable. That is there is an assignment α such that both α modified mapping $x \mapsto 0$, and α modified mapping $x \mapsto 1$, falsify some polynomials in S depending on x. Let α_0 and α_1 be such assignments. In the case of twin variables α_0 also sets $\bar{x} \mapsto 1$ and α_1 also sets $\bar{x} \mapsto 0$.

Let the polynomials falsified by α_0 and α_1 resp. be f and f'. That is

$$f(\alpha_0) \neq 0 \text{ and } f'(\alpha_1) \neq 0 .$$

Since S is x-saturated, $f_{x\mapsto 0} f_{x\mapsto 1} \equiv 0$ and $f'_{x\mapsto 0} f'_{x\mapsto 1} \equiv 0$. Hence it must be that

$$f(\alpha_1) = 0 \text{ and } f'(\alpha_0) = 0 .$$

Again by the assumption on S being x-saturated,

$$(f + f')_{x\mapsto 0}(f + f')_{x\mapsto 1} \equiv 0 ,$$

That is

$$(f + f')(\alpha_0) \cdot (f + f')(\alpha_1) = 0 .$$

On the other hand, this last equality is not possible since

$$(f + f')(\alpha_0) = f(\alpha_0) + f'(\alpha_0) \neq 0$$

and similarly $(f + f')(\alpha_1) \neq 0$. □

Lemma 5.11. *In the context of H the Boolean axioms, for every set of weighted polynomials S and every variable x, there is a $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ derivation from S of a set of polynomials S' which is x-saturated.*

Proof. For a polynomial f, recall that $f_{x\mapsto 0}$ is the evaluation of f in $x = 0$ and, in the case of twin variables, the restriction also sets $\bar{x} = 1$ (resp. for $f_{x\mapsto 1}$).

Suppose we have a set of weighted polynomials S and a variable x. We construct a sequence of weighted polynomials $S_0, S_1, \ldots$ to find the set S'. We start with $S_0 = S$, then we want S_{i+1} to be derivable from S_i using the rules of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$, and moreover, in S_{i+1} we added some new polynomial non-dependent on x.

There are two possible types of inference we could use to go from S_i to S_{i+1}.

First type. For each $i \geq 0$, if there is an $[f, w] \in S_i$ depending on x and s.t. $f_{x\mapsto 0} f_{x\mapsto 1} \not\equiv 0$, choose any $[f, w]$ and let

$$S_{i+1} = \big(S_i \setminus \{[f, w]\}\big) \cup \{[f_{x\mapsto 0} f_{x\mapsto 1}, w],\ [f_{x\mapsto 0} f_{x\mapsto 1} + f, w]\} .$$

The derivation of S_{i+1} from S_i, by substituting $[f, w]$ with the weighted polynomials $[f_{x\mapsto 0} f_{x\mapsto 1}, w]$ and $[f_{x\mapsto 0} f_{x\mapsto 1} + f, w]$, is justified by:

$$\dfrac{\dfrac{[f, w]}{[f f_{x\mapsto 0} f_{x\mapsto 1}, w] \quad [f(f_{x\mapsto 0} f_{x\mapsto 1} + 1), w]}\ \textsc{prod}}{[f_{x\mapsto 0} f_{x\mapsto 1}, w] \quad [f_{x\mapsto 0} f_{x\mapsto 1} + f, w]}\ \equiv$$

where the last $\equiv$ holds since $f f_{x\mapsto 0} f_{x\mapsto 1} \equiv f_{x\mapsto 0} f_{x\mapsto 1}$ (Lemma 5.7). Notice that, with this substitution, we have obtained the weighted polynomial $[f_{x\mapsto 0} f_{x\mapsto 1}, w]$ where the variable x does not appear (and hence clearly not depending on x) and it is not equivalent to 0 since the condition to obtain S_{i+1} is that $f_{x\mapsto 0} f_{x\mapsto 1} \not\equiv 0$. We used the assumption that f depends on x to ensure the polynomial $f_{x\mapsto 0} f_{x\mapsto 1}$ in the conclusions is new and it is not equivalent to the polynomial f in the premises.

Notice that on the weighted polynomial derived still possibly depending on x, $[f_{x\mapsto 0} f_{x\mapsto 1} + f, w]$ we cannot apply a PROD, since

$$\begin{aligned}(f_{x\mapsto 0} f_{x\mapsto 1} + f)_{x\mapsto 0}(f_{x\mapsto 0} f_{x\mapsto 1} + f)_{x\mapsto 1} &= (f_{x\mapsto 0} f_{x\mapsto 1} + f_{x\mapsto 0})(f_{x\mapsto 0} f_{x\mapsto 1} + f_{x\mapsto 1}) \\ &\equiv 4 f_{x\mapsto 0} f_{x\mapsto 1} \\ &= 0\end{aligned}$$

Therefore, eventually we cannot apply PROD anymore and we need to start summing polynomials.

Second type. If there are weighted polynomials $[f, w], [g, w'] \in S_i$ depending on x, with

$$(f_{x\mapsto 0} + g_{x\mapsto 0})(f_{x\mapsto 1} + g_{x\mapsto 1}) \not\equiv 0 ,$$

and $w' \geq w > 0$, non-deterministically choose two of them. First substitute $[g, w']$ by $[g, w]$ and $[g, w' - w]$, and then let

$$\begin{aligned}S_{i+1} = \big(S_i \setminus \{[f, w],\ [g, w]\}\big) \cup \{&[(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1}, w],\ [fg, 2w], \\ &[(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1} + f + g, w]\} .\end{aligned}$$

We can obtain S_{i+1} from S_i using first the SUM rule to infer $[f + g, w]$ and $[fg, 2w]$ and then use the PROD rule on $[f + g, w]$ as we did in the previous case on a single polynomial.

$$\dfrac{\dfrac{[f\,, w] \quad [g\,, w]}{[f+g\,, w] \quad [fg\,, 2w]}\ \textsc{sum}}{[(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1}\,, w] \quad [(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1}+f+g\,, w] \quad [fg\,, 2w]}\ \textsc{prod}\ \&\equiv$$

Doing this substitution we obtain a polynomial where the variable x does not appear and it is not equivalent to 0 since the condition to obtain S_{i+1} is that $(f_{x\mapsto 0}+g_{x\mapsto 0})(f_{x\mapsto 1}+g_{x\mapsto 1}) \not\equiv 0$.

Notice that the polynomials introduced still containing x could be summed with other ones, or in some cases we could even apply a PROD to $[fg\,, 2w]$. As before we cannot apply the first type of inference to $[(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1}+f+g\,, w]$, since this would give a polynomial equivalent to 0.

If we cannot transform S_i into S_{i+1} in either of the two ways above we stop the process.

This sequence of transformations must be finite and the last element S_ℓ will be x-saturated by construction. The process must be finite since otherwise the sequence given by

$$\sigma(i) = \sum_{\substack{\alpha:\ X\to\mathbb{F}_2\\ H\text{-compatible}}} \operatorname{cost}(\alpha, S_i^+)$$

for S_i^+ the part of S_i depending on x, would give a sequence of strictly decreasing natural numbers. Indeed, all the $\sigma(i)$s are natural numbers because we are using the rules of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$, so, no negative weight could appear in any S_i and $\operatorname{cost}(\alpha, S_i^+) \geq 0$. To show that $\sigma(i+1) < \sigma(i)$ it is sufficient to notice that, by the soundness of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$, for every H-compatible $\alpha:\ X \to \mathbb{F}_2$,

$$\operatorname{cost}(\alpha, S_{i+1}) = \operatorname{cost}(\alpha, S_i)\ ,$$

which implies that

$$\operatorname{cost}(\alpha, S_{i+1}^+) + \operatorname{cost}(\alpha, \{[h\,, w]\}) = \operatorname{cost}(\alpha, S_i^+)\ ,$$

for some polynomial $h \not\equiv 0$, not depending the variable x and with $w > 0$. Hence

$$\sigma(i+1) + \sum_{\substack{\alpha:\ X\to\mathbb{F}_2\\ H\text{-compatible}}} \operatorname{cost}(\alpha, \{[h\,, w]\}) = \sigma(i)\ ,$$

and

$$\sum_{\substack{\alpha:\ X\to\mathbb{F}_2\\ H\text{-compatible}}} \operatorname{cost}(\alpha, \{[h, w]\}) > 0$$

because $h \not\equiv 0$ and $w > 0$. Therefore, $\sigma(i+1) < \sigma(i)$. And the sequence must be finite. □

We now show how to obtain the completeness (Theorem 5.1), essentially iterating the process from the previous lemma on all variables one by one.

Proof of Theorem 5.1. Let $X = \{x_1, \dots, x_n\}$. By Lemma 5.11, from F we can derive in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ a set S_1 x_1-saturated. By Lemma 5.10 S_1 is x_1-fixable. Let $S_1 = S_1^+ \cup S_1^-$, where S_1^+ is the part of S_1 depending on x_1 and S_1^- the part of S_1 not depending on x_1.

Again, by Lemma 5.11, from S_1^- we can derive in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ a set S_2 x_2-saturated. By Lemma 5.10 S_2 is x_2-fixable. This gives a decomposition $S_2 = S_2^+ \cup S_2^-$, where S_2^- are the weighted polynomials in S_2 not depending on x_2 (and x_1). Continuing in this way by all the variables of X one by one we arrive at a set S_n, where the weighted polynomials in S_n^- are just constants, i.e. $S_n^- \approx \{[1\,, w]\}$ for some $w \in \mathbb{N}$.

To show that $w = \operatorname{cost}_H(F)$ it is enough to show that $\bigcup_{j\in[n]} S_j^+$ is satisfiable by a H-compatible assignment. Let $\alpha:\ X \to \mathbb{F}_2$ be an arbitrary H-compatible assignment. Since S_n is x_n-fixable there is a way to modify α in x_n to get a H-compatible assignment satisfying all S_n^+. Let this assignment be α_n. Suppose we obtained a H-compatible assignment α_i satisfying $\bigcup_{j\geq i} S_j^+$, since S_{i-1} is x_{i-1}-fixable, there is a way to modify α_i in x_{i-1} to satisfy all S_{i-1}^+. Let this assignment be α_{i-1}. Since the polynomials in $\bigcup_{j\geq i} S_j^+$ only contained the variables $x_i \dots, x_n$, the assignment α_{i-1} continues to satisfy $\bigcup_{j\geq i} S_j^+$. We continue this way until we get an assignment α_1 satisfying all $\bigcup_{j\in[n]} S_j^+$. Thus proving that it must have been that $w = \operatorname{cost}_H(F)$. □

An implementation in Python of the algorithm showing the completeness is available as supplemental material.

We conclude this section with an alternative proof of completeness that does not make use of the PROD rule. This proof is only valid for $\mathbb{F}_2$ and, unlike the previous completeness proof, it is not generalized in Section 7. We also show that the number of SUM applications needed for the completeness is quadratic in the sum of the weights of the original polynomials.

Given two polynomials f and g, applying twice the SUM rule we can infer:

$$\begin{array}{c|l} [f,w] \quad [g,w] & \\ \hline [f+g,w] \quad [fg,2w] & \text{SUM} \\ \hline [f+g,w] \quad [fg,w] \quad [fg,w] & \approx \\ \hline [f+g+fg,w] \quad [fg,w] & \text{SUM} \end{array} \tag{7}$$

where in the second application of SUM we also infer $(f+g)fg$, which is omitted since it is always zero.

When polynomials encode Boolean formulas, the polynomial fg is satisfied by α if f *or* g are satisfied by α, and $f+g+fg$ is satisfied if f *and* g are satisfied. Hence, fg encode $f \vee g$ and $f+g+fg$ encode $f \wedge g$. Therefore, this sequence of two applications of the SUM rule proves

$$f,\ g \vdash f \vee g,\ f \wedge g\ .$$

Notice also that (on the contrary to SUM and PROD rules), the concatenation of two SUM rules in eq. (7), and the structural inference rules preserve or decrease (when the polynomial zero is generated) the sum of the weights of the clauses. This allows us to bound the number of rule applications in the following theorem. We stress that bounding the number of rules application does not imply a non-trivial upper bound size of the $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ derivation. The polynomials in the derivation might have exponential number of terms.

Theorem 5.12 *(completeness of SUM). Given F a set of weighted polynomials over $\mathbb{F}_2[X]$ and the set of Boolean axioms as hard constraints H, there is a $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ derivation of $[1, \mathrm{cost}_H(F)]$ from F that uses only the SUM rule and the structural rules.*

Moreover, the number of rule applications is bounded by $2(\sum_{[f,w]\in F} w)^2$.

Proof. The proof is by induction on the value of $\mathrm{cost}_H(F)$. If this cost is zero, the empty derivation suffices to infer the desired polynomial.

When $\mathrm{cost}_H(F) > 0$, we start by deriving the *conjunction* of all the polynomials in F. In $\mathbb{F}_2$ we have $f+g+fg \equiv (f+1)(g+1)+1$, which can be interpreted as a kind of de Morgan equivalence $f \wedge g \equiv \overline{\overline{f} \vee \overline{g}}$. Therefore, after $2(|F|-1)$ applications of the SUM rule we can get the *conjunction* of all polynomials in F, that using the de Morgan laws we could write as:

$$[1 + \prod_{[f,w]\in F} (f+1),\ \min_{[f,w]\in F} w]\ . \tag{8}$$

This weighted polynomial comes together with a set of polynomials F' satisfying

$$\mathrm{cost}_H(F') = \mathrm{cost}_H(F) - \min_{[f,w]\in F} w\ ,$$

by the soundness of SUM, and

$$\sum_{[f,w]\in F'} w \leq \min_{[f,w]\in F} w + \sum_{[f,w]\in F'} w\ ,$$

by the preservation of the sum of weights mentioned above.

Since $\mathrm{cost}_H(F) > 0$, for every α, there exists at least one $[f, w] \in F$ such that $f(\alpha) = 1$. Hence, for every α, polynomial (8) gets value one. Therefore, this polynomial is equivalent to the polynomial one. By induction, from F' we can derive $[1, \mathrm{cost}_H(F')]$ with at most $2(\sum_{[f,w]\in F'} w)^2$ applications of the SUM rule. Concatenating both derivations, we get $[1, \mathrm{cost}_H(F)]$ from F in at most $2(|F|-1) + 2(\sum_{[f,w]\in F'} w)^2$ SUM steps, i.e. in $2(\sum_{[f,w]\in F} w)^2$ steps because $|F| \leq \sum_{[f,w]\in F} w$. □

6. Tseitin formulas

We exemplify the algorithm for the completeness on *Tseitin formulas*, although also Example 4.3 was found using a variation of the algorithm for the completeness.

First, we recall what are Tseitin formulas. For this section, consider fixed a graph $G = (V, E)$ with $|V| = 2n+1$ and Boolean variables $x_{v,w}$ for each $\{v, w\} \in E$. For $v \in V$, let $N(v) = \{w \in V : \{v, w\} \in E\}$. The *Tseitin formula* on G is a CNF formula expressing that in each vertex $v \in V$ the parity of the variables x_e for the edges incident to v is 1, that is Tseitin(G) is the CNF

$$\bigcup_{v\in V} \Big\{ \bigoplus_{w\in N(v)} x_{v,w} = 1 \pmod 2 \Big\}\ , \tag{9}$$

where $\bigoplus_{w\in N(v)} x_{v,w} = 1 \pmod 2$ is encoded as a set of clauses. For instance, if $N(v) = \{w_1, w_2, w_3\}$, then $\bigoplus_{w\in N(v)} x_{v,w} + 1 \pmod 2$ is

$$\begin{aligned}&\{x_{v,w_1}, x_{v,w_2}, x_{v,w_3}\},\ \{\neg x_{v,w_1}, \neg x_{v,w_2}, x_{v,w_3}\},\\ &\{x_{v,w_1}, \neg x_{v,w_2}, \neg x_{v,w_3}\},\ \{\neg x_{v,w_1}, x_{v,w_2}, \neg x_{v,w_3}\}\ . \end{aligned}\tag{10}$$

Since V has an odd size, Tseitin(G) is unsatisfiable.

Consider first the natural encoding of eq. (9) as polynomials. That is consider the set of variables $X = \{x_{v,w} : \{v, w\} \in E\}$ and L_v be the polynomial $\sum_{w\in N(v)} x_{v,w} + 1$ in $\mathbb{F}_2[X]$.

It is well known that $\mathsf{PC}_{\mathbb{F}_2}$ is able to refute $\{L_v : v \in V\}$ in linear size. In $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ we prove more.

Proposition 6.1. *There is a linear size derivation in* $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ *of* $[1, c]$ *from* $\{[L_v, 1] : v \in V\}$, *where c is the number of connected components of odd size in G. In particular, if G is connected,* $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ *proves that* $\{[L_v, 1] : v \in V\}$ *is minimally unsatisfiable, i.e. it is possible to satisfy all polynomials in it except one.*

Proof. We show how to infer $[1, 1]$ from $\{[L_v, 1] : v \in V\}$ via the saturation process, when G is connected. For the saturation process, the order in which we saturate the variables is not important.

At each intermediate saturation step ℓ there is a set of weighted polynomials $\mathcal{S}_\ell$ that we have to saturate. The set $\mathcal{S}_\ell$ has the form $\{[L_{S_1}, 1], \dots, [L_{S_m}, 1]\}$, where $S_1, \dots, S_m$ form a partition of V and $L_{S_i} = \sum_{v\in S_i} L_v$. Moreover, we already saturated w.r.t. all the variables $x_{v,w}$ with v, w in the same S_i.

At the beginning of the saturation process, we have the partition of V consisting of all the singletons: $\{\{v\} : v \in V\}$.

Suppose then we are at an intermediate step ℓ of the saturation. We have a set $\mathcal{S}_\ell = \{[L_{S_1}, 1], \dots, [L_{S_m}, 1]\}$ and we want to saturate w.r.t. $x_{v,w}$. By the inductive assumption, $\{v, w\}$ is not an internal edge of any of the sets S_is. Hence there are exactly two distinct sets S_i and S_j with $v \in S_i$ and $w \in S_j$. That is, to saturate $\mathcal{S}_\ell$ w.r.t. $x_{v,w}$ is enough to saturate $\mathcal{S}' = \{[L_{S_i}, 1], [L_{S_j}, 1]\}$. We follow the procedure from Lemma 5.11.

Fact 1. For every <u>linear</u> polynomial L depending on x,

$$L_{x\mapsto 0} L_{x\mapsto 1} = L_{x\mapsto 0}(L_{x\mapsto 0} + 1) \equiv 0\ .$$

By Fact 1, the only possibility is to SUM L_{S_i} and L_{S_j}. That is from $\mathcal{S}'$ we obtain

$$\mathcal{S}'' = \{[L_{S_i} + L_{S_j}, 1], [L_{S_i} L_{S_j}, 2]\}\ .$$

Now, $L_{S_i} + L_{S_j} \equiv L_{S_i\cup S_j}$ does not contain variables $x_{v',w'}$ with $v', w' \in S_i \cup S_j$. In particular it does not contain $x_{v,w}$. To continue the saturation process, the only possibility would be to do a PROD on $L_{S_i} L_{S_j}$, but this produces the polynomial $L_{S_i,x=0} L_{S_j,x=0} L_{S_i,x=1} L_{S_j,x=1} \equiv 0$ by Fact 1. Therefore $\mathcal{S}''$ is saturated w.r.t. $x_{v,w}$. And so is the multi-set

$$\{[L_{S_k}, 1] : k \neq i, j\} \cup \{[L_{S_i} + L_{S_j}, 1], [L_{S_i} L_{S_j}, 2]\}\ .$$

The part of this set not depending on $x_{v,w}$ is

$$\mathcal{S}_{\ell+1} = \{[L_{S_k}, 1] : k \neq i, j\} \cup \{[L_{S_i} + L_{S_j}, 1]\}\ .$$

Notice that $[L_{S_i} L_{S_j}, 2]$ is not in $\mathcal{S}_{\ell+1}$ since it depends on $x_{v,w}$. The set $\mathcal{S}_{\ell+1}$ is the one we want to saturate at the next step for some other variable. During the saturation process we obtain coarser and coarser partitions of V and, at the end of the whole process, we obtain $\{[L_V, 1]\}$. To conclude we just need to observe that $L_V \equiv |V| \equiv 1$. □

To show that $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ and weighted Resolution are incomparable, we need to consider Tseitin(G) encoded as a set of polynomials using the twin variables encoding from Section 2.2. Assume all the initial polynomials of this encoding to have weight 1. From this system of polynomials is still easy to derive $[1, c]$ in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ where c is the number of connected components of odd size in G. Such derivations can be found using the saturation process, provided we use the natural heuristic of preferentially taking the SUM of two weighted polynomials $[f, w]$ and $[g, w]$ when $fg \equiv 0$. The intuitive reason behind this heuristic is that in such a SUM the number of polynomials in conclusion decreases and we do not introduce a polynomial of higher degree. Under this heuristic it is immediate to see that the saturation process will essentially reconstruct the polynomials $\{[L_v, 1] : v \in V(G)\}$. Indeed, take for instance the twin variables encoding of the set of clauses in eq. (10), that is

$$\begin{aligned}S = \{&[\bar{x}_{v,w_1}\bar{x}_{v,w_2}\bar{x}_{v,w_3}, 1], [x_{v,w_1}x_{v,w_2}\bar{x}_{v,w_3}, 1],\\ &[\bar{x}_{v,w_1}x_{v,w_2}x_{v,w_3}, 1], [x_{v,w_1}\bar{x}_{v,w_2}x_{v,w_3}, 1]\}\ .\end{aligned}$$

We have that the product of any two of the polynomials is divisible by the polynomial $x_{v,w_i}\bar{x}_{v,w_i} \equiv 0$ for some i. Therefore applying the SUM rule on S, eventually we obtain

$$[\bar{x}_{v,w_1}\bar{x}_{v,w_2}\bar{x}_{v,w_3} + x_{v,w_1}x_{v,w_2}\bar{x}_{v,w_3} + \bar{x}_{v,w_1}x_{v,w_2}x_{v,w_3} + x_{v,w_1}\bar{x}_{v,w_2}x_{v,w_3}, 1]$$

which is equivalent under the $\equiv$ relation to $[L_v, 1]$.

7. Polynomial calculus for MaxSAT (general case)

In this section, we adapt the definition of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ from $\mathbb{F}_2$ to an arbitrary finite field $\mathbb{F}_q$.

The general forms of the PROD and SUM rules are:

$$\frac{[\,f\,,\,w\,]}{[\,fg\,,\,w\,]\quad [\,f(g^{q-1}-1)\,,\,w\,]}\qquad (\textsc{prod}_q)$$

$$\frac{[\,f\,,\,w\,]\quad [\,g\,,\,w\,]}{[\,f+g\,,\,w\,]\quad [\,fg\,,\,w\,]\quad [\,f\left((f+g)^{q-1}-1\right)\,,\,w\,]}\qquad (\textsc{sum}_q)$$

for all $f,g\in\mathbb{F}_q[X]$ and $w\in\mathbb{Z}$. The $\textsc{prod}_q$ rule is strongly sound since for every assignment $\alpha:\ X\to\mathbb{F}_q$ if $f(\alpha)=0$ the cost of the premise is 0 and so is the cost of the conclusion. If $f(\alpha)\neq 0$, then either $g(\alpha)=0$ or $g(\alpha)\neq 0$, but in this latter case then $g^{q-1}(\alpha)=1$. The soundness of the $\textsc{sum}_q$ rule is analogous.

Notice that the PROD and SUM rules for $\mathbb{F}_2[X]$ are special cases of $\textsc{prod}_q$ and $\textsc{sum}_q$ (modulo the structural rules, the fold-unfold and the $\equiv$). Indeed for $q=2$,

$$f(g-1)=f(g+1)\ ,$$

and

$$f((f+g)-1)\equiv fg\ .$$

Using the $\textsc{prod}_q$ and $\textsc{sum}_q$, it is immediate to generalize the definition of $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ from weighted polynomials with coefficients in $\mathbb{F}_2$ to weighted polynomials with coefficients in $\mathbb{F}_q$.

Definition 7.1 *($\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}$ and $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$).* The systems are defined as in Definition 4.1 except that all the substitution rules now refer to polynomials in $\mathbb{F}_q[X]$ and instead of the PROD/SUM we use the $\textsc{prod}_q$ and $\textsc{sum}_q$. We call the resulting systems $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}$ and $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$.

Similar to the case of $\mathbb{F}_2$, we have that the $\textsc{sum}_q$ rule is redundant in $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$:

Lemma 7.2. *Using weights in $\mathbb{Z}$ and the structural rules, the $\textsc{prod}_q$ rule can simulate the $\textsc{sum}_q$ rule.*

Proof.

$$\begin{array}{cl}
[\,f\,,\,w\,]\quad [\,g\,,\,w\,] & \\
\hline
[\,f\,,\,w\,]\quad [\,g\,,\,w\,]\quad [\,f+g\,,\,w\,]\quad [\,f+g\,,\,-w\,] & \approx \\
\hline
[\,f\,,\,w\,]\quad [\,g\,,\,w\,]\quad [\,f+g\,,\,w\,]\quad [\,f(f+g)\,,\,-w\,]\quad [\,(f^{q-1}-1)(f+g)\,,\,-w\,] & \textsc{prod}_q \\
\hline
[\,f\,,\,w\,]\quad [\,g\,,\,w\,]\quad [\,f+g\,,\,w\,]\quad [\,f(f+g)\,,\,-w\,]\quad [\,g(f^{q-1}-1)\,,\,-w\,] & \equiv \\
\hline
[\,f\,,\,w\,]\quad [\,gf\,,\,w\,]\quad [\,g(f^{q-1}-1)\,,\,w\,]\quad [\,f+g\,,\,w\,]\quad [\,f(f+g)\,,\,-w\,]\quad [\,g(f^{q-1}-1)\,,\,-w\,] & \textsc{prod}_q \\
\hline
[\,f\,,\,w\,]\quad [\,gf\,,\,w\,]\quad [\,f+g\,,\,w\,]\quad [\,f(f+g)\,,\,-w\,] & \approx \\
\hline
[\,f(f+g)\,,\,w\,]\quad [\,f((f+g)^{q-1}-1)\,,\,w\,]\quad [\,gf\,,\,w\,]\quad [\,f+g\,,\,w\,]\quad [\,f(f+g)\,,\,-w\,] & \textsc{prod}_q \\
\hline
[\,f((f+g)^{q-1}-1)\,,\,w\,]\quad [\,gf\,,\,w\,]\quad [\,f+g\,,\,w\,] & \approx
\end{array}\qquad\square$$

As for the case of $\mathbb{F}_2$ we prove the soundness and completeness of $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}$ and $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$.

Theorem 7.3 *(soundness). Given $F=\{[\,f_1\,,\,w_1\,],\dots,[\,f_m\,,\,w_m\,]\}$ where $f_i\in\mathbb{F}_q[X]$ and a set of polynomials $H\subseteq\mathbb{F}_q[X]$, if there is a $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$ derivation of $[\,1\,,\,w\,]$ from F (and H as hard constraints), then $\mathrm{cost}_H(F)\geq w$.*

Proof. The argument is a simple generalization of the proof of Theorem 4.4. Let $L_0,L_1,L_2,\dots,L_s$ be a $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$ derivation of $[\,1\,,\,w\,]$, i.e. L_s contains $[\,1\,,\,w\,]$, $L_0=F$ and each L_{i+1} is obtained from L_i applying the $\textsc{prod}_q$, the $\textsc{sum}_q$ substitution rules, the fold-unfold equivalence or the H-equivalence. We have that $\mathrm{cost}_H(L_s)\geq w$ since $[\,1\,,\,w\,]\in L_s$ and all the other weighted polynomials in L_s have non-negative weights. Hence, to prove the statement is enough to show that for each i, $\mathrm{cost}_H(L_{i+1})=\mathrm{cost}_H(L_i)$. We prove something slightly stronger, that for each H-consistent $\alpha:\ X\to\mathbb{F}_q$, $\mathrm{cost}(\alpha,L_{i+1})=\mathrm{cost}(\alpha,L_i)$. This follows immediately from the comments we already made on the soundness of the various substitution rules. $\square$

We show the completeness in two cases. The case of Boolean axioms, that is $X=\{x_1,\dots,x_n\}$ with $H=\{x_i^2-x_i\ :\ i\in[n]\}$, and the case of Boolean axioms with twin variables $X=\{x_1,\dots,x_n,\bar{x}_1,\dots,\bar{x}_n\}$ and $H=\{x_i^2-x_i,\ x_i+\bar{x}_i-1\ :\ i\in[n]\}$.

Theorem 7.4 *(completeness for Boolean variables). Given F a multiset of weighted polynomials in $\mathbb{F}_q[X]$, there is a $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}$ derivation of* $[\,1\,,\,\mathrm{cost}_H(F)\,]$ *from F, and the set of Boolean axioms as hard constraints.*

The argument is a generalization of the argument we saw in Section 5.

As for the special case of $\mathbb{F}_2$ we first define formally when a polynomial depends on a variable x.

Definition 7.5 *(dependence, general case).* Let $H \subseteq \mathbb{F}_q[X]$ be a finite set. We say that a polynomial f does not *depend* on a variable x w.r.t. H if there exists a polynomial g not containing x (and also $\bar{x}$ in the case of twin variables) such that $f \equiv_H g$.

We have a characterization of the notion of dependence similar to Proposition 5.4.

Proposition 7.6. *Let f be a polynomial in $\mathbb{F}_q[X]$, x a variable and $H \subseteq \mathbb{F}_q[X]$ be a finite set. The following are equivalent*

1. *f does not depend on x w.r.t. H;*
2. *$f \equiv_H f_{x\mapsto 0} \equiv_H f_{x\mapsto 1}$;*
3. *$f \equiv_H f^{q-2} f_{x\mapsto 0} f_{x\mapsto 1}$.*

Proof. Since H is fixed thorough this proof we omit it, but implicitly every $\equiv$ is with respect to the fixed H. We prove that item 1 implies item 2, item 2 implies item 3, and item 3 implies item 1.

Item 1 implies item 2: If f does not depend on x, then by definition there exist g not containing x such that $f \equiv g$, hence restricting by $x = 0$ and $x = 1$ we get $f \equiv f_{x\mapsto 0} \equiv f_{x\mapsto 1}$.

Item 2 implies item 3: If $f \equiv f_{x\mapsto 0} \equiv f_{x\mapsto 1}$, then $f^{q-2} f_{x\mapsto 0} f_{x\mapsto 1} \equiv f^q \equiv f$.

Item 3 implies item 1: If $f^{q-2} f_{x\mapsto 0} f_{x\mapsto 1} \equiv f$ we can evaluate in $x = 0$ and $x = 1$ to get

$$f^{q-1}_{x\mapsto 0} f_{x\mapsto 1} \equiv f_{x\mapsto 0} \quad \text{and} \quad f^{q-1}_{x\mapsto 1} f_{x\mapsto 0} \equiv f_{x\mapsto 1}\ .$$

Multiplying the first equation by $f_{x\mapsto 0}$ and the second by $f_{x\mapsto 1}$, we get

$$f_{x\mapsto 0} f_{x\mapsto 1} \equiv f^2_{x\mapsto 0} \quad \text{and} \quad f_{x\mapsto 0} f_{x\mapsto 1} \equiv f^2_{x\mapsto 1}\ .$$

Hence summing we get $(f_{x\mapsto 0} - f_{x\mapsto 1})^2 \equiv 0$. Which implies $f_{x\mapsto 0} \equiv f_{x\mapsto 1}$. Moreover

$$f \equiv x f_{x\mapsto 1} + (1-x) f_{x\mapsto 0} \equiv x f_{x\mapsto 1} + (1-x) f_{x\mapsto 1} \equiv f_{x\mapsto 1}.$$

Hence $f \equiv f_{x\mapsto 0} \equiv f_{x\mapsto 1}$. And in particular f is equivalent to a polynomial without x. □

Notice that, similarly to the case of $\mathbb{F}_2$, by Proposition 7.6, to check whether a polynomial f depends or not on a variable x it is enough to check whether $f \equiv_H f_{x\mapsto 0} \equiv_H f_{x\mapsto 1}$. Section 3 discusses effective ways to check this in the cases when $H = \{x_i^2 - x_i : i \in [n]\}$ and the twin variables $H = \{x_i^2 - x_i, x_i + \bar{x}_1 - 1 : i \in [n]\}$.

The notion of x-fixable set also generalizes immediately from $\mathbb{F}_2$ to $\mathbb{F}_q$.

Definition 7.7 *(x-fixable set, general case).* Let $H \subseteq \mathbb{F}_q[X]$ be a finite set, $x \in X$ and S a set of weighted polynomials. The set S is *x-fixable* if every H-compatible assignment $\alpha : X \to \mathbb{F}_q$ can be modified in x to a H-compatible assignment satisfying all weighted polynomials in S that depend on x.

The notion of saturated set is analogous to the case of $\mathbb{F}_2$.

Definition 7.8 *(x-saturated set, general case).* Let S a set of polynomials in $\mathbb{F}_q[X]$, $x \in X$ and $H = \{x^2 - x : x \in X\}$ or H the Boolean axioms with twin variables. The set S is *x-saturated w.r.t. H* if

- for all $f \in S$ depending on x, $f_{x\mapsto 0} f_{x\mapsto 1} \equiv_H 0$, and
- for all $f, g \in S$ depending on x, $(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1} \equiv_H 0$.

As for $\mathbb{F}_2$ we have that a set being saturated w.r.t. x implies it being x-fixable.

Lemma 7.9. *When H the Boolean axioms or H are Boolean axioms with the twin variables, if S is x-saturated w.r.t. H, then S is x-fixable w.r.t. H.*

Proof. The argument is identical to the proof of Lemma 5.10. □

The following lemma is an adaptation of Lemma 5.11.

Lemma 7.10. *In the context of H the Boolean axioms, for every set of weighted polynomials S and every variable x, there is a* $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}$ *derivation from S of a set of polynomials S' which is saturated w.r.t. x.*

Proof. The proof is analogous to the argument for Lemma 5.11.

For a polynomial f, recall that $f_{x\mapsto 0}$ is the evaluation of f in $x = 0$ and, in the case of twin variables, the restriction also sets $\bar{x} = 1$ (resp. for $f_{x\mapsto 1}$).

Suppose we have a set of weighted polynomials S and a variable x. We construct a sequence of weighted polynomials $S_0, S_1, \dots$ to find the set S'. We start with $S_0 = S$, then we want S_{i+1} to be derivable from S_i using the rules of $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}$, and moreover, in S_{i+1} we added some new polynomial non-dependent on x. The way to obtain S_{i+1} from S_i is the following. For each $i \geq 0$, if there is an $[f, w] \in S_i$ depending on x and s.t. $f_{x\mapsto 0} f_{x\mapsto 1} \not\equiv 0$, non-deterministically choose one of such $[f, w]$ and let

$$S_{i+1} = \big(S_i \setminus \{[f, w]\}\big) \cup \{[f_{x\mapsto 0} f_{x\mapsto 1}, w],\ [f f_{x\mapsto 0}^{q-1} f_{x\mapsto 1}^{q-1} - f, w]\}\ .$$

The derivation of S_{i+1} from S_i, by substituting $[f, w]$ with the weighted polynomials $[f_{x\mapsto 0} f_{x\mapsto 1}, w]$ and $[f(f_{x\mapsto 0} f_{x\mapsto 1})^{q-1} - f, w]$, is justified by:

$$\begin{array}{c l}
[f, w] & \\ \hline
[f f^{q-2} f_{x\mapsto 0} f_{x\mapsto 1}, w] \quad [f((f^{q-2} f_{x\mapsto 0} f_{x\mapsto 1})^{q-1} - 1), w] & \textsc{prod}_q \\ \hline
[f f^{q-2} f_{x\mapsto 0} f_{x\mapsto 1}, w] \quad [f^q (f^{q-2} f_{x\mapsto 0} f_{x\mapsto 1})^{q-1} - f, w] & \equiv \\ \hline
[f^{q-1} f_{x\mapsto 0} f_{x\mapsto 1}, w] \quad [f (f^{q-1} f_{x\mapsto 0} f_{x\mapsto 1})^{q-1} - f, w] & \equiv \\ \hline
[f_{x\mapsto 0} f_{x\mapsto 1}, w] \quad [f (f_{x\mapsto 0} f_{x\mapsto 1})^{q-1} - f, w] & \equiv
\end{array}$$

where the last $\equiv$ holds since $f^{q-1} f_{x\mapsto 0} f_{x\mapsto 1} \equiv f_{x\mapsto 0} f_{x\mapsto 1}$. Notice that, with this substitution, we have obtained the weighted polynomial $[f_{x\mapsto 0} f_{x\mapsto 1}, w]$ where the variable x does not appear (and hence clearly not depending on x) and it is not equivalent to 0 since the condition to obtain S_{i+1} is that $f_{x\mapsto 0} f_{x\mapsto 1} \not\equiv 0$. We used the assumption that f depends on x to ensure the polynomial $f_{x\mapsto 0} f_{x\mapsto 1}$ in the conclusions is new and it is not equivalent to the polynomial f in the premises.

If there are $[f, w], [g, w'] \in S_i$ depending on x, with $f \not\equiv g$,

$$(f_{x\mapsto 0} + g_{x\mapsto 0})(f_{x\mapsto 1} + g_{x\mapsto 1}) \not\equiv 0\ ,$$

and $w' \geq w > 0$, non-deterministically choose two of them. First substitute $[g, w']$ by $[g, w]$ and $[g, w' - w]$, and then let

$$\begin{aligned}
S_{i+1} = \big(&S_i \setminus \{[f, w],\ [g, w]\}\big) \\
&\cup \{\ [(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1}, w], \\
&\qquad [(f+g)(f+g)_{x\mapsto 0}^{q-1}(f+g)_{x\mapsto 1}^{q-1} - f - g, w] \\
&\qquad [fg, w], \\
&\qquad [f(f+g)^{q-1} - f, w]\ \}\ .
\end{aligned}$$

We can obtain S_{i+1} from S_i using first the $\textsc{sum}_q$ rule and then use the $\textsc{prod}_q$ rule on $[f + g, w]$ as we did in the previous case on a single polynomial.

$$\begin{array}{c l}
[f, w] \quad [g, w] & \\ \hline
[f+g, w] \quad [fg, w] \quad [f(f+g)^{q-1} - f, w] & \textsc{sum}_q \\ \hline
[(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1}, w] \quad [(f+g)(f+g)_{x\mapsto 0}^{q-1}(f+g)_{x\mapsto 1}^{q-1} - f - g, w] & \textsc{prod}_q \ \& \equiv \\
[fg, w] \quad [f(f+g)^{q-1} - f, w] &
\end{array} \tag{11}$$

Doing this substitution we obtain a polynomial where the variable x does not appear and it is not equivalent to 0 since the condition to obtain S_{i+1} is that $(f_{x\mapsto 0} + g_{x\mapsto 0})(f_{x\mapsto 1} + g_{x\mapsto 1}) \not\equiv 0$.

If we cannot transform S_i into S_{i+1} in either of the two ways we stop the process. With the same argument of Lemma 5.11 this sequence of transformations must be finite and the last element S_ℓ will be saturated w.r.t. x. □

We can now prove Theorem 7.4.

Proof of Theorem 7.4. Let $X = \{x_1, \dots, x_n\}$. First saturate F w.r.t. x_1. By Lemma 7.10, from F in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ we can derive a set S_1 saturated w.r.t. x_1. By Lemma 7.9 S_1 is x_1-fixable. Let $S_1 = S_1^+ \cup S_1^-$, where S_1^+ is the part of S_1 depending on x_1 and S_1^- the part of S_1 not depending on x_1.

Saturate S_1^- w.r.t. x_2. Again, by Lemma 7.10, from S_1^- we can derive in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ an set S_2 saturated w.r.t. x_2. By Lemma 7.9 S_2 is x_2-fixable. This gives a decomposition $S_2 = S_2^+ \cup S_2^-$, where S_2^- are the weighted polynomials in S_2 not depending on x_2 (and x_1). Continuing in this way by all the variables of X one by one we arrive at a set S_n, where the weighted polynomials in S_n^- are just constants, i.e.

$$S_n^- \approx \{[c_j, w_j] : j \in J\}$$

for some w_js in $\mathbb{N}$ and non-zero constants c_j. Now notice that for non-zero constant c and a weight w, the rule

$$\frac{[c, w]}{[1, w]}$$

is a special case of the $\textsc{prod}_q$ rule together with an application of $\approx$, indeed

$$\cfrac{\cfrac{[c, w]}{[c \cdot c^{-1}, w] \quad [c((c^{-1})^{q-1} - 1), w]}\ \textsc{prod}_q}{[1, w]}\ \approx$$

since $(c^{-1})^{q-1} = 1$ because by assumption $c \neq 0$.

That is from S_ℓ we can derive using the special case of the $\textsc{prod}_q$ above $\{[1, \sum_{j\in J} w_j]\}$

To show that $\sum_{j\in J} w_j = \mathrm{cost}_H(F)$ it is enough to show that $\bigcup_{j\in[n]} S_j^+$ is satisfiable by a H-compatible assignment. The argument to prove this is identical to the one for the completeness in the case of $\mathbb{F}_2$. □

Notice, that as in the case of $\mathbb{F}_2$ we proved something stronger, that $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}$ is complete also if we restrict the $\textsc{prod}_q$ rule to be of the form

$$\frac{[f, w]}{[f_{x\mapsto 0} f_{x\mapsto 1}, w] \quad [f f_{x\mapsto 0}^{q-1} f_{x\mapsto 1}^{q-1} - f, w]} \tag{12}$$

and

$$\frac{[c, w]}{[1, w]}$$

for arbitrary non-zero constant c.

We conclude this section with an example of application of the saturation process for polynomials over $\mathbb{F}_3$. The example we give is a special case of a type of *covering* principle.

Given q a power of a prime and a hyper-graph $\mathcal{H} = (V, E)$ we consider the principle asserting that $\mathcal{H}$ admits an edge cover C where for each $v \in V$, $|\{e \in C : v \in e\}| = 1 \pmod q$. Depending on $\mathcal{H}$ this might be satisfiable or unsatisfiable, and in the case it is unsatisfiable we are interested in seeing the maximum number of constraints we can simultaneously satisfy.

To encode this principle we use Boolean variables x_e for each hyperedge $e \in E$ whose meaning is the truth value of "*the edge e belongs to the edge cover C*". The principle is then encoded as the following set of weighted polynomials over $\mathbb{F}_q[\{x_e : e \in E\}]$:

$$C_q(\mathcal{H}) = \left\{ [\sum_{e \ni v} x_e - 1, 1] \ : \ v \in V \right\}$$

together with the set of hard constraints $H = \{x_e^2 - x_e \ : \ e \in E\}$.

From this general construction we consider the example of $C_3(\mathcal{H})$ for the hypergraph $\mathcal{H}$ with vertices [9] and hyper-edges

$$\{\{1,2,3\}, \{3,4,5\}, \{5,6,7\}, \{7,8,9\}\}\ .$$

That is

$$\begin{aligned} C_3(\mathcal{H}) = \{&[x-1, 1], [x-1, 1], [x+y-1, 1],\\ &[y-1, 1], [y+z-1, 1],\\ &[z-1, 1], [z+v-1, 1],\\ &[v-1, 1], [v-1, 1]\}\ , \end{aligned}$$

where we use x, y, z, v to denote the four hyper-edges of $\mathcal{H}$, that is x is an alias for $x_{\{1,2,3\}}$, y for $x_{\{3,4,5\}}$, z for $x_{\{5,6,7\}}$, and v for $x_{\{7,8,9\}}$.

We use the saturation process from the proof of Theorem 7.4 to show that $\mathrm{cost}_H(C_3(\mathcal{H}))$ is 2, when H is the set of the Boolean axioms.

The saturation order we choose is x, y, z, v. The multiset $C_3(\mathcal{H})$ is not saturated w.r.t. x. Simple calculations show that the only possibility is to use the $\textsc{sum}_3$ and then $\textsc{prod}_3$ rule on $[x-1, 1]$ and $[x+y-1, 1]$ following the general scheme of eq. (11). Labeling $x-1$ as f, and $x+y-1$ as g, $(f+g)_{x\mapsto 0}(f+g)_{x\mapsto 1} = -y$. Then

$$\begin{aligned}(f+g)^2_{x\mapsto 0}(f+g)^2_{x\mapsto 1} &= y\\ (f+g)(f+g)^2_{x\mapsto 0}(f+g)^2_{x\mapsto 1} - f - g &= -xy + x + y - 1\\ fg &= xy - x - y + 1\\ f(f+g)^2 - f &= 0\ .\end{aligned}$$

Given the previous equalities, we obtain the multiset

$$\begin{aligned}S_1 = \{&[-y, 1],\ [-xy + x + y - 1, 1],\ [xy - x - y + 1, 1],\\ &[x-1, 1],\\ &[y-1, 1], [y+z-1, 1],\\ &[z-1, 1], [z+v-1, 1],\\ &[v-1, 1], [v-1, 1]\}\ .\end{aligned}$$

This multiset S_1 is saturated w.r.t. x, given that

$$\begin{aligned}(-xy + x + y - 1)_{x\mapsto 1} &= 0\\ (-xy + x + y - 1) + (xy - x - y + 1) &= 0\\ ((x-1) + (-xy + x + y - 1))_{x\mapsto 1} &= 0\\ ((x-1) + (xy - x - y + 1))_{x\mapsto 1} &= 0\ .\end{aligned}$$

To proceed with the saturation process then we only need consider the part of S_1 not dependent on x, that is

$$\begin{aligned}S_1' = \{&[-y, 1],\\ &[y-1, 1], [y+z-1, 1],\\ &[z-1, 1], [z+v-1, 1],\\ &[v-1, 1], [v-1, 1]\}\ .\end{aligned}$$

We now saturate S_1' w.r.t. y. The multiset S_1' is not already saturated w.r.t. y, indeed we could use the $\textsc{sum}_3$ and then $\textsc{prod}_3$ rule on $[-y, 1]$ and $[y-1, 1]$, or alternatively on $[y-1, 1]$ and $[y+z-1, 1]$, or alternatively on $[-y, 1]$ and $[y+z-1, 1]$. We choose to use the $\textsc{sum}_3$ and then $\textsc{prod}_3$ rule on $[-y, 1]$ and $[y-1, 1]$ again following the general scheme of eq. (11). Since $-y + y - 1 = -1$, we already have a contradiction. After the $\textsc{prod}_3$ this gives the wighted polynomial $[1, 1]$. Also,

$$\begin{aligned}-1 \cdot (-1)^2_{y\mapsto 0}(-1)^2_{y\mapsto 1} - (-1) &= 0\\ -y(y-1) &= 0\\ (-y)(-1)^2 - (-y) &= 0\ ,\end{aligned}$$

this gives the multiset

$$\begin{aligned}S_2 = \{&[1, 1],\\ &[y+z-1, 1],\\ &[z-1, 1], [z+v-1, 1],\\ &[v-1, 1], [v-1, 1]\}\ .\end{aligned}$$

The multiset S_2 is saturated w.r.t. y because we cannot have a $\textsc{prod}_3$ on $y + z - 1$ to get a polynomial without y. To proceed with the saturation process then we only need consider the part of S_2 not dependent on y, that is

$$\begin{aligned}S_2' = \{&[1, 1],\\ &[z-1, 1], [z+v-1, 1],\end{aligned}$$

$$[v-1,1],[v-1,1]\}\ .$$

The multiset S_2' is not saturated w.r.t. z indeed we can use the $\textsc{sum}_3$ and then $\textsc{prod}_3$ rule on $[z-1,1]$ and $[z+v-1,1]$. Notice that this situation is analogous to the case of the polynomials $x-1$ and $x+y-1$ that we had when saturating w.r.t. x. Therefore the result will be equivalent. So we obtain the multiset

$$\begin{aligned}S_3=\{&[1,1],\\&[-v,1],\ [-zv+z+v-1,1],\ [zv-z-v+1,1],\\&[v-1,1],[v-1,1]\}\ .\end{aligned}$$

The multiset S_3 is saturated w.r.t. z. To proceed with the saturation process then we only need to consider the part of S_3 not dependent on z, that is

$$\begin{aligned}S_3'=\{&[1,1],\\&[-v,1],\\&[v-1,1],[v-1,1]\}\ .\end{aligned}$$

The multiset S_3' is not saturated w.r.t. v indeed we can use the $\textsc{sum}_3$ and then $\textsc{prod}_3$ rule on $[v-1,1]$ and $[-v,1]$ which is analogous to the situation of the saturation of the variable y. This gives the multiset

$$\begin{aligned}S_4=\{&[1,2],\\&[v-1,1]\}\ .\end{aligned}$$

The multiset S_4 is saturated w.r.t. v and this concludes the saturation process in the iterative way used in Theorem 7.4. That is, we just showed that $\mathrm{cost}_H(\mathcal{C}_3(\mathcal{H}))=2$, for the given hypergraph $\mathcal{H}$ and given as hard constraints the Boolean axioms.

8. Connections of weighted polynomial calculus with other MaxSAT systems

The system $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ with twin variables is at least as strong as $\mathbb{N}$-weighted Resolution and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ is at least as strong as $\mathbb{Z}$-weighted Resolution (aka Sherali-Adams and Circular Resolution [26,21]). Moreover, all the four systems can be separated either using Tseitin formulas or the Pigeonhole principle, as shown in Fig. 2.

Proposition 8.1. *The system $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ (with twin variables) is exponentially stronger than $\mathbb{N}$-weighted Resolution and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ (with twin variables) is exponentially stronger than $\mathbb{Z}$-weighted Resolution.*

Proof. It is well known that PC (with the twin variable encoding, see Section 2.2) p-simulates Resolution [28]. The same simulation, with the weights properly added to the polynomials, gives that $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ with twin variables p-simulates $\mathbb{N}$-weighted Resolution. The same happens for weights in $\mathbb{Z}$, giving that $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ with twin variables p-simulates $\mathbb{Z}$-weighted Resolution, which is equivalent to Sherali-Adams and Circular Resolution.

Moreover PC is exponentially stronger than Resolution, given that Tseitin formulas are easy for PC (over $\mathbb{F}_2$) while exponentially hard for Resolution [30] and Sherali-Adams [35]. Therefore, Tseitin formulas also exponentially separate $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ from $\mathbb{Z}$-weighted Resolution (Sherali-Adams) and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ from $\mathbb{N}$-weighted Resolution. □

Proposition 8.2. *The system $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ is exponentially stronger than $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ and $\mathbb{Z}$-weighted Resolution is exponentially stronger than $\mathbb{N}$-weighted Resolution.*

Proof. The simulations are trivial since proofs using weights in $\mathbb{N}$ are just special cases of weights in $\mathbb{Z}$. On the other hand the Pigeonhole principle is easy in $\mathbb{Z}$-weighted Resolution [26] and exponentially hard for PC [36], and therefore for $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$. The Pigeonhole principle can be proved polynomially in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ since this system polynomially simulates $\mathbb{Z}$-weighted Resolution. □

Indeed both Proposition 8.1 and 8.2 follow from the fact that the systems $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ (with twin variables) and $\mathbb{Z}$-weighted Resolution are incomparable: in the first system Tseitin is easy to prove while the Pigeonhole principle is hard, and it is the opposite for the latter system. Both the Pigeonhole principle and Tseitin have short proofs in $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$.

The relationships between $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}/\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ and the aforementioned systems are summarized in Fig. 2.

We recall that Fig. 2 can also be read in the context of propositional proof systems (for SAT). Indeed, all the MaxSAT systems in Fig. 2 can be seen as propositional proof systems, if the weights of the initial clauses/polynomials are *not* part of the input but part of the proof and to refute we just want to derive one instance of the empty clause or the polynomial 1. In this setting weighted Resolution is the same as Resolution and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ is the same as PC over $\mathbb{F}_2$.

Analogous simulations as the ones in Fig. 2 hold also for $\mathsf{wPC}_{\mathbb{F}_q,\mathbb{N}}/\mathsf{wPC}_{\mathbb{F}_q,\mathbb{Z}}$ using a Tseitin principle modulo q instead of the usual one.

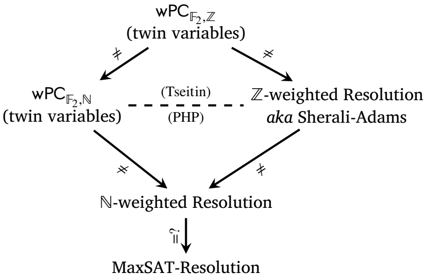
$\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$
(twin variables)
$\neq$
$\neq$
$\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$
(twin variables)
(Tseitin)
(PHP)
$\mathbb{Z}$-weighted Resolution
aka Sherali-Adams
$\neq$
$\neq$
$\mathbb{N}$-weighted Resolution
$\overset{?}{=}$
MaxSAT-Resolution

Fig. 2. $P \to Q$ means that P is at least as strong as Q. A dashed line means the two systems are incomparable.

9. Conclusions

We generalized Polynomial Calculus to the context of MaxSAT for polynomials with coefficients in a finite field. This involves extending the rules of Polynomial Calculus to have additional conclusions and applying the rules by replacing premises with conclusions, to make the system sound for MaxSAT. We showed its completeness via a saturation process. The resulting proof system may be used for SAT or for MaxSAT. The system $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{N}}$ is stronger than MaxSAT Resolution, and $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ is stronger than $\mathbb{Z}$-weighted Resolution (aka Sherali-Adams).

The proposed calculus might be useful in the context of parity or mod p reasoning when we want to optimize the minimum number of unsatisfiable clauses or polynomials. Natural areas of parity/XOR reasoning are logical cryptanalysis, circuit verification, and bounded model checking. As an example, we show how our saturation algorithm is able to prove efficiently that Tseitin formulas are minimally unsatisfiable, in contrast to what happens in the case of Resolution and Sherali-Adams. Another possible direction of application might be in the context of graph-coloring where elements of the field $\mathbb{F}_p$ are used to directly encode the colors.

We conclude with a couple of open problems:

1. Polynomial Calculus is degree-automatable, in the sense that bounded degree proofs can be found efficiently (in time $n^{O(d)}$, where d is the degree). It is open whether $\mathsf{wPC}_{\mathbb{F}_2,\mathbb{Z}}$ is degree-automatable as-well.
2. It is also open whether there is a suitable notion of polynomial calculus for MaxSAT using polynomials over infinite fields, or even over generic finite rings.

CRediT authorship contribution statement

Ilario Bonacina: Conceptualization, Investigation, Writing – original draft, Writing – review & editing. **Maria Luisa Bonet:** Conceptualization, Investigation, Writing – original draft, Writing – review & editing. **Jordi Levy:** Conceptualization, Investigation, Software, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was supported by the grant numbers PID2019-109137GB-C21, PID2019-109137GB-C22, IJC2018-035334-I, PID2022-138506NB-C21, and PID2022-138506NB-C22 funded by AEI. This work was partially done while Maria Luisa Bonet and Jordi Levy were visiting the Simons Institute for the Theory of Computing during the Spring of 2023.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.artint.2024.104208.

References

[1] I. Bonacina, M.L. Bonet, J. Levy, Polynomial calculus for MaxSAT, in: Proc. of the 26th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'23), 2023, pp. 5:1–5:17.

[2] K. Pipatsrisawat, A. Darwiche, On the power of clause-learning SAT solvers as resolution engines, Artif. Intell. 175 (2) (2011) 512–525, https://doi.org/10.1016/j.artint.2010.10.002.

[3] A. Atserias, J.K. Fichte, M. Thurley, Clause-learning algorithms with many restarts and bounded-width resolution, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2009, pp. 114–127.

[4] A. Biere, M. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability - Second Edition, Frontiers in Artificial Intelligence and Applications, vol. 336, IOS Press, 2021.

[5] A. Ignatiev, A. Morgado, J. Marques-Silva, On tackling the limits of resolution in SAT solving, in: Proc. of the 20th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'17), 2017, pp. 164–183.

[6] M.L. Bonet, S. Buss, A. Ignatiev, A. Morgado, J. Marques-Silva, Propositional proof systems based on maximum satisfiability, Artif. Intell. 300 (2021) 103552, https://doi.org/10.1016/j.artint.2021.103552.

[7] C. Ansótegui, J. Levy, Reducing SAT to Max2SAT, in: Proc. of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21), 2021, pp. 1367–1373.

[8] B. Buchberger, A theoretical basis for the reduction of polynomials to canonical forms, SIGSAM Bull. 10 (3) (1976) 19–29, https://doi.org/10.1145/1088216.1088219.

[9] M. Brickenstein, A. Dreyer, PolyBoRi: a framework for Groebner-basis computations with Boolean polynomials, J. Symb. Comput. 44 (9) (2009) 1326–1345, https://doi.org/10.1016/j.jsc.2008.02.017, Effective Methods in Algebraic Geometry.

[10] J.C. 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 '02, Association for Computing Machinery, New York, NY, USA, 2002, pp. 75–83.

[11] J.A. De Loera, J. Lee, S. Margulies, S. Onn, Expressing combinatorial problems by systems of polynomial equations and Hilbert's Nullstellensatz, Comb. Probab. Comput. 18 (4) (2009) 551–582, https://doi.org/10.1017/S0963548309009894.

[12] J.A. De Loera, J. Lee, P.N. Malkin, S. Margulies, Computing infeasibility certificates for combinatorial problems through Hilbert's Nullstellensatz, J. Symb. Comput. 46 (11) (2011) 1260–1283, https://doi.org/10.1016/j.jsc.2011.08.007.

[13] J.A. De Loera, S. Margulies, M. Pernpeintner, E. Riedl, D. Rolnick, G. Spencer, D. Stasi, J. 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, ACM, 2015, pp. 133–140.

[14] D. Kaufmann, A. Biere, M. Kauers, From DRUP to PAC and back, in: 2020 Design, Automation & Test in Europe Conference & Exhibition, DATE 2020, Grenoble, France, March 9-13, 2020, 2020, pp. 654–657.

[15] D. Kaufmann, A. Biere, Nullstellensatz-proofs for multiplier verification, in: Computer Algebra in Scientific Computing - 22nd International Workshop, CASC 2020, Linz, Austria, September 14-18, 2020, Proceedings, 2020, pp. 368–389.

[16] D. Kaufmann, A. Biere, M. Kauers, Verifying large multipliers by combining SAT and computer algebra, in: 2019 Formal Methods in Computer Aided Design, FMCAD 2019, San Jose, CA, USA, October 22-25, 2019, 2019, pp. 28–36.

[17] D. Kaufmann, P. Beame, A. Biere, J. 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'22), 2022, pp. 1431–1436.

[18] M. Clegg, J. Edmonds, R. Impagliazzo, Using the Groebner basis algorithm to find proofs of unsatisfiability, in: G.L. Miller (Ed.), Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, ACM, 1996, pp. 174–183.

[19] M.L. Bonet, J. Levy, F. Manyà, A complete calculus for Max-SAT, in: Proc. of the 9th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'06), 2006, pp. 240–251.

[20] M.L. Bonet, J. Levy, F. Manyà, Resolution for max-SAT, Artif. Intell. 171 (8–9) (2007) 606–618, https://doi.org/10.1016/j.artint.2007.03.001.

[21] M.L. Bonet, J. Levy, Equivalence between systems stronger than resolution, in: L. Pulina, M. Seidl (Eds.), Theory and Applications of Satisfiability Testing – SAT 2020, Springer International Publishing, Cham, 2020, pp. 166–181.

[22] I. Bonacina, M.L. Bonet, J. Levy, Weighted, circular and semi-algebraic proofs, J. Artif. Intell. Res. 79 (2024) 447–482, https://doi.org/10.1613/JAIR.1.15075.

[23] J. Larrosa, E. Rollón, Augmenting the power of (partial) MaxSAT resolution with extension, in: Proc. of the 34th Nat. Conf. on Artificial Intelligence (AAAI'20), 2020, pp. 1561–1568.

[24] J. Larrosa, E. Rollon, Towards a better understanding of (partial weighted) maxsat proof systems, in: Proc. of the 23rd Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'20), 2020, pp. 218–232.

[25] E. Rollon, J. Larrosa, Proof complexity for the maximum satisfiability problem and its use in SAT refutations, J. Log. Comput. 32 (7) (2022) 1401–1435, https://doi.org/10.1093/logcom/exac004.

[26] A. Atserias, M. Lauria, Circular (yet sound) proofs, in: M. Janota, I. Lynce (Eds.), Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings, in: Lecture Notes in Computer Science, vol. 11628, Springer, 2019, pp. 1–18.

[27] C. Ansótegui, M.L. Bonet, J. Levy, F. Manyà, The logic behind weighted CSP, in: M.M. Veloso (Ed.), IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007, Hyderabad, India, 2007, pp. 32–37.

[28] S. Buss, D. Grigoriev, R. Impagliazzo, T. Pitassi, Linear gaps between degrees for the polynomial calculus modulo distinct primes, J. Comput. Syst. Sci. 62 (2) (2001) 267–289, https://doi.org/10.1006/jcss.2000.1726.

[29] S.A. Cook, R.A. Reckhow, The relative efficiency of propositional proof systems, J. Symb. Log. 44 (1) (1979) 36–50, https://doi.org/10.2307/2273702.

[30] A. Urquhart, Hard examples for resolution, J. ACM 34 (1) (1987) 209–219, https://doi.org/10.1145/7531.8928.

[31] S.F. de Rezende, M. Lauria, J. Nordström, D. Sokolov, The power of negative reasoning, in: V. Kabanets (Ed.), 36th Computational Complexity Conference (CCC 2021), in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 200, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 40:1–40:24.

[32] D. Sokolov, (Semi)algebraic proofs over $\{\pm 1\}$ variables, in: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, ACM, 2020, pp. 78–90.

[33] S. Mouli, Polynomial calculus sizes over the Boolean and Fourier bases are incomparable, arXiv:2403.03933, 2024.

[34] Y. Filmus, E.A. Hirsch, A. Riazanov, A. Smal, M. Vinyals, Proving unsatisfiability with hitting formulas, in: V. Guruswami (Ed.), 15th Innovations in Theoretical Computer Science Conference (ITCS 2024), in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 287, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024, pp. 48:1–48:20.

[35] D. Grigoriev, Linear lower bound on degrees of positivstellensatz calculus proofs for the parity, Theor. Comput. Sci. 259 (1–2) (2001) 613–622, https://doi.org/10.1016/S0304-3975(00)00157-2.

[36] A. Razborov, Lower bounds for the polynomial calculus, Comput. Complex. 7 (4) (1998) 291–324, https://doi.org/10.1007/s000370050013.