Polynomial Analysis of Modular Arithmetic

Thomas Seed[1], Chris Coppins[1], Andy King[1(✉)], and Neil Evans[2]

[1] University of Kent, Canterbury CT2 7NZ, UK
a.m.king@kent.ac.uk
[2] AWE Aldermaston, Reading RG7 4PR, UK

Abstract. The modular polynomial abstract domain, MPAD, is proposed, whose invariants are systems of polynomial equations that hold modulo a power of 2. Its domain operations are founded on a closure operation, but unlike conventional polynomial abstractions, MPAD satisfies the ascending chain condition, can model both positive and negative polynomial guards, and can infer invariants previously out of reach.

Keywords: abstract interpretation · modular arithmetic · polynomial invariants

1 Introduction

One step in the evolution of a numeric abstract domain is when the domain, originally conceived for idealised, arbitrary-precision arithmetic, is adapted to machine arithmetic to better suit its working environment. This adaption is more often a leap than a step because the domain operations typically need to be fundamentally reimagined to model modular arithmetic. It has taken more than two decades for each of the classical abstract domains of ranges [7,15], difference constraints [9] and linear equalities [20], to be adjusted to a modular setting, as realised in, respectively, sign agnostic range analysis [11], modular differences [12] and linear equalities modulo a power of two [27]. The tenor of these works is that operating over modular integers is not a restriction, but rather the natural domain for deriving invariants over fixed-width integers, which are the norm in mainstream programming languages.

Modular Polynomial Abstract Domain. For inferring polynomial invariants, one might be forgiven for considering the additional complexity of modular arithmetic to be an irritation, justified only by the desire to faithfully model machine integers and avoid missing invariants. In this paper we challenge this view by demonstrating how Modular Polynomial Abstract Domain (MPAD) can simplify the discovery of polynomial equalities. Contrary to non-modular approaches [4,8,17,22,23,25,29–31], MPAD is a finite lattice. The finiteness of modular polynomials has been observed before [33], and exploited in a backwards analysis [33] over programs equipped with polynomial assignment, non-deterministic assignment and negative guards (discussed in Sect. 7). Our work takes modular polynomials in a new direction, literally forwards, enabling MPAD to be combined [5]

M. V. Hermenegildo and J. F. Morales (Eds.): SAS 2023, LNCS 14284, pp. 508–539, 2023.
https://doi.org/10.1007/978-3-031-44245-2_22

with classic numeric domains [24]. Like [33], MPAD obviates the need to specify the shape of an invariant up-front (in a template) [29,31], or limit the syntactic form of the program [17,18,23], or drop high-degree polynomials [30].

Closure of Modular Systems. Fundamental to MPAD is the concept of a closed polynomial system. A system of polynomials is closed if it cannot be further augmented with polynomials without restricting its solution set. Ensuring a closed representation is essential to ensure that entailment of a given constraint can always be checked. Morever, mirroring a construction used for the Octagon domain [24], we demonstrate that join and projection can be calculated, without omitting polynomials that actually hold, when they are applied to closed systems. It follows that MPAD can infer all modular polynomial invariants for programs with polynomial and non-deterministic assignments, and non-deterministic branching. Though preserved by join and projection, closedness is lost when intersecting two polynomial systems to compute their meet. To resolve this, we present a divide-and-conquer algorithm for computing closure, thus ensuring a closed representation throughout the analysis.

Expressiveness. MPAD can model positive and negative polynomial guards, that is, $\mathsf{assume}\ (p = 0)$ and $\mathsf{assume}\ (p \neq 0)$ statements where p is a polynomial. Support for negative guards is a direct consequence of working with fixed-width integers: an integer assumes a non-zero value if and only if one of its bits is set, a property that can be expressed in MPAD. The finiteness of MPAD also allows a best transformer [28] to be mechanically calculated. For instance, the best transformer for the 3-bit bitvector operation $x \mathbin{\&} y = z$ is the system S:

$$
\begin{array}{ll}
xy^3 + xy^2 + 5yz^2 + 2xy + 5z^2 + 2z, & xyz + 7xz^2 + 7yz^2 + z^3, \\
xz^3 + xz^2 + 5z^3 + 6xz + 5z^2 + 6z, & y^2z + 7yz^2 + yz + 7z^2, \\
yz^3 + 5yz^2 + z^3 + 2yz + 5z^2 + 2z, & 4xy + 4z, \\
x^2y + 5xy^2 + 6xz^2 + 6yz^2 + 6z^3 + 5xy + 3z^2, & 4xz + 4z, \\
x^2z + 7xz^2 + 5xz + 3z^2, & 4yz + 4z
\end{array}
$$

where each polynomial $p \in S$ is satisfied by every assignment of the form $x, y \in \{0, \ldots, 7\}$ and $z = (x \& y) \mod 8$. This, and other best abstractions, can only be calculated [28] because MPAD satisfies the ascending chain condition.

Contributions. To summarise, this paper makes the following contributions:

- We introduce closure for MPAD, showing that it is preserved by join and projection but must be re-established after meet to retain all invariants;
- We present a divide-and-conquer algorithm for computing closure, introducing reductions and shortcuts that simplify its calculation;
- We show how redundant calculation can be removed from the algorithm (of Buchberger for modular polynomials [2]) which sits behind closure;
- We show that using MPAD in forwards analysis can derive invariants that cannot be derived with existing domains (because of its support for guards).

Roadmap. Section 2 introduces MPAD, providing the minimum of detail for following the example of Sect. 6.5. The domain operations of MPAD are built atop of Gröbner bases, which are introduced in Sect. 3. (The detail of Sect. 3.5 can be skipped on first reading since it is only necessary for Sect. 6.6.) Sect. 4 explains how join can be calculated in terms of variable elimination and Gröbner bases. Section 5 introduces covers of polynomial systems, providing an algorithm for computing them. It also shows how meet can be reduced to closure. Section 6 presents correctness and precision results for MPAD over polynomial programs, and concludes with an illustrative example. Section 7 reviews related work and Sect. 8 concludes.

2 Modular Polynomial Abstract Domain

This section abstractly specifies MPAD, and its domain operations, with minimal mathematical machinery. The problems of how to finitely represent the elements of MPAD and compute meet, join and projection are deferred to later sections.

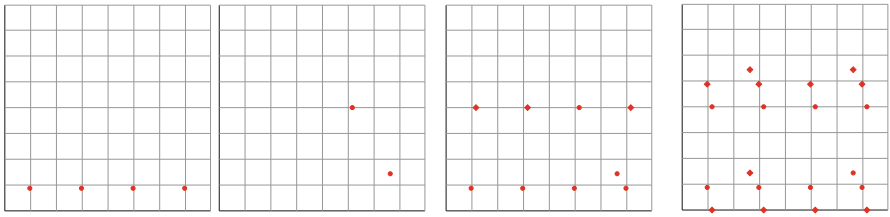
2.1 Modular Arithmetic

Let $\omega \geq 1$, $m = 2^\omega$ and $\mathbb{Z}_m = \{0, \ldots, m-1\}$ be an abstraction of machine arithmetic over ω-bit integers [26,27]. The relation $\equiv_m \subseteq \mathbb{Z} \times \mathbb{Z}$ is defined by $x \equiv_m y$ if there exists $k \in \mathbb{Z}$ such that $x - y = km$. Atop, the operation $\cdot \pmod m : \mathbb{Z} \to \mathbb{Z}_m$ is defined $x \pmod m = y$ where $y \in \mathbb{Z}_m$ uniquely satisfies $x \equiv_m y$. The unary operation $- : \mathbb{Z}_m \to \mathbb{Z}_m$ and the dyadic operations $+, \cdot : \mathbb{Z}_m \times \mathbb{Z}_m \to \mathbb{Z}_m$ are then defined: $-x = (\hat{-}x) \pmod m$, $x + y = (x \mathbin{\hat{+}} y) \pmod m$ and $x \cdot y = (x \mathbin{\hat{\cdot}} y) \pmod m$ where $\hat{-}, \hat{+}, \hat{\cdot}$ denote the classical operations over $\mathbb{Z}$. If $x \in \mathbb{Z}_m$ then $y \in \mathbb{Z}_m$ is a multiplicative inverse of x if $x \cdot y = 1$. Note that $x \in \mathbb{Z}_m$ has a multiplicative inverse iff it is odd, in which case the inverse is unique. In particular, if $\omega > 1$ then $\mathbb{Z}_m$ is not a field, since 2 has no multiplicative inverse.

2.2 Polynomials

Let $\boldsymbol{x} = \langle x_1, \ldots, x_d \rangle$ be a vector of variables. A monomial over $\boldsymbol{x}$ is an expression $\boldsymbol{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdots x_d^{\alpha_d}$ where $\boldsymbol{\alpha} = \langle \alpha_1, \ldots, \alpha_d \rangle \in \mathbb{N}^d$. A term over $\boldsymbol{x}$ is an expression $t = c\boldsymbol{x}^{\boldsymbol{\alpha}}$ where $c \in \mathbb{Z}_m$ and $\boldsymbol{x}^{\boldsymbol{\alpha}}$ is a monomial. A polynomial over $\boldsymbol{x}$ is an expression $t_1 + \cdots + t_s$ where each t_i is a term over $\boldsymbol{x}$, the case $s = 0$ corresponding to the 0 polynomial. The set of polynomials over $\boldsymbol{x}$ is denoted $\mathbb{Z}_m[\boldsymbol{x}]$.

A polynomial $p = t_1 + \cdots + t_s$ is normalised if either $s = 0$ or else for all $t_i = c_i \boldsymbol{x}^{\boldsymbol{\alpha}_i}$ and $t_j = c_j \boldsymbol{x}^{\boldsymbol{\alpha}_j}$ it holds that $c_i \neq 0$ and if $i \neq j$ then $\boldsymbol{\alpha}_i \neq \boldsymbol{\alpha}_j$. By repeatedly combining the coefficients of terms with equal monomials, and deleting terms with coefficient 0, a polynomial can be transformed into a normalised form. Two polynomials are considered equal if they have equal normal forms, up to the ordering of terms. If $c\boldsymbol{x}^{\boldsymbol{\alpha}}$ is a term then $\mathsf{vars}(c\boldsymbol{x}^{\boldsymbol{\alpha}}) = \{x_i \mid \alpha_i > 0\}$, which is extended to polynomials by $\mathsf{vars}(p) = \bigcup_{t \in p} \mathsf{vars}(t)$.



(a) $\gamma_{\boldsymbol{x}}(Q_1)$ (b) $\gamma_{\boldsymbol{x}}(Q_2)$ (c) $\gamma_{\boldsymbol{x}}(\langle Q_1' \rangle_{\boldsymbol{x}} \cap \langle Q_2 \rangle_{\boldsymbol{x}})$ (d) $\gamma_{\boldsymbol{x}}(\langle Q_1 \rangle_{\boldsymbol{x}} \cap \langle Q_2 \rangle_{\boldsymbol{x}})$

Fig. 1. Dyadic join with and without closure

For $p \in \mathbb{Z}_m[\boldsymbol{x}]$, $d = |\boldsymbol{x}|$ and $\boldsymbol{a} \in \mathbb{Z}_m^d$ let $[\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a})$ denote evaluating p at $\boldsymbol{a}$ by substituting each a_i for x_i in p, and calculating the resulting arithmetical expression. Through this definition, a set of polynomials in $\mathbb{Z}_m[\boldsymbol{x}]$ is a symbolic description of a set of points, interpreted by $\gamma_{\boldsymbol{x}}$ as follows:

Definition 1. *The concretisation map* $\gamma_{\boldsymbol{x}} : \wp(\mathbb{Z}_m[\boldsymbol{x}]) \to \wp(\mathbb{Z}_m^d)$ *where* $d = |\boldsymbol{x}|$ *is defined:* $\gamma_{\boldsymbol{x}}(P) = \{\boldsymbol{a} \in \mathbb{Z}_m^d \mid [\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a}) = 0 \text{ for all } p \in P\}$

The set of points $\gamma_{\boldsymbol{x}}(P)$ is the solution (or zero) set of the set P of polynomials over $\boldsymbol{x}$. For a single $p \in \mathbb{Z}_m[\boldsymbol{x}]$, let $\gamma_{\boldsymbol{x}}(p) = \gamma_{\boldsymbol{x}}(\{p\})$.

Example 1. Let $\boldsymbol{x} = \langle x, y \rangle$ and $Q_1, Q_2 \subseteq \mathbb{Z}_{256}[\boldsymbol{x}]$ where

$$Q_1 = \{\, 4x + 132, y + 228 \,\} \quad Q_2 = \left\{ \begin{array}{lll} x^2 + x + 123y + 130, & xy + 108y + 128, \\ 2x + 23y + 54, & y^2 + 82y, \quad 128y \end{array} \right\}$$

The solutions sets $\gamma_{\boldsymbol{x}}(Q_1)$ and $\gamma_{\boldsymbol{x}}(Q_2)$ are plotted as points in $[0, 255]^2$ in Fig. 1(a) and Fig. 1(b) respectively. Here, the grid lines represent increments of 32. Although Q_1 is linear it has 4 solutions, namely (31, 28), (95, 28), (159, 28) and (223, 28), because $31 \cdot 4 \equiv_{256} 95 \cdot 4 \equiv_{256} 159 \cdot 4 \equiv_{256} 223 \cdot 4 \equiv_{256} 124 \equiv_{256} -132$.

2.3 Closure

Suppose $P \subseteq \mathbb{Z}_m[\boldsymbol{x}]$, $p \in \mathbb{Z}_m[\boldsymbol{x}]$ and $\gamma_{\boldsymbol{x}}(P) \subseteq \gamma_{\boldsymbol{x}}(p)$. Then $\gamma_{\boldsymbol{x}}(P \cup \{p\}) = \gamma_{\boldsymbol{x}}(P)$, thus P can be augmented with p without restricting its solution set. This is the intuition behind the following definition:

Definition 2. *The operator* $\uparrow_{\boldsymbol{x}} : \wp(\mathbb{Z}_m[\boldsymbol{x}]) \to \wp(\mathbb{Z}_m[\boldsymbol{x}])$ *is defined by:* $\uparrow_{\boldsymbol{x}} P = \{p \in \mathbb{Z}_m[\boldsymbol{x}] \mid \gamma_{\boldsymbol{x}}(P) \subseteq \gamma_{\boldsymbol{x}}(p)\}$

The following result collects fundamental properties of $\uparrow_{\boldsymbol{x}}$. The first three together imply that $\uparrow_{\boldsymbol{x}}$ is a closure operator on $\langle \wp(\mathbb{Z}_m[\boldsymbol{x}]), \subseteq \rangle$. The fourth implies that $\uparrow_{\boldsymbol{x}}$ constructs a canonical representation of a system of polynomials. The fifth shows that the canonical representation preserves the solution set.

Proposition 1. *The operator* $\uparrow_{\boldsymbol{x}}$ *satisfies the following: (1)* $P \subseteq \uparrow_{\boldsymbol{x}} P$ *(extensive); (2) if* $P_1 \subseteq P_2$ *then* $\uparrow_{\boldsymbol{x}} P_1 \subseteq \uparrow_{\boldsymbol{x}} P_2$ *(monotonic); (3)* $\uparrow_{\boldsymbol{x}} \uparrow_{\boldsymbol{x}} P = \uparrow_{\boldsymbol{x}} P$ *(idempotent); (4)* $\gamma_{\boldsymbol{x}}(P_1) = \gamma_{\boldsymbol{x}}(P_2)$ *iff* $\uparrow_{\boldsymbol{x}} P_1 = \uparrow_{\boldsymbol{x}} P_2$*; (5)* $\gamma_{\boldsymbol{x}}(\uparrow_{\boldsymbol{x}} P) = \gamma_{\boldsymbol{x}}(P)$.

The closure operator $\uparrow_{\boldsymbol{x}}$ yields a canonical representation of a given set of polynomials, yet the representation is not finite. The concept of a basis is introduced, which serves as the starting point for a compact, finite representation:

Definition 3. *If* $B \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ *then* $\langle B\rangle_{\boldsymbol{x}} = \{\sum_{i=1}^{s} u_i p_i \mid s \in \mathbb{N}, p_i \in B, u_i \in \mathbb{Z}_m[\boldsymbol{x}]\}$

The set of polynomials $\langle B\rangle_{\boldsymbol{x}}$ is an ideal in that it is closed under addition with a polynomial from B and multiplication with an arbitrary polynomial (not necessarily drawn from B). The ideal $\langle B\rangle_{\boldsymbol{x}}$ is said to be generated by B, which is called the basis. In particular the solutions of $\langle B\rangle_{\boldsymbol{x}}$ are those of B itself and the sets found by applying closure are ideals themselves:

Lemma 1. *(1)* $\gamma_{\boldsymbol{x}}(\langle B\rangle_{\boldsymbol{x}}) = \gamma_{\boldsymbol{x}}(B)$ *and (2) If* $P = \uparrow_{\boldsymbol{x}} P$ *then* $P = \langle P\rangle_{\boldsymbol{x}}$.

Generating P from itself is enough to show that P is an ideal, but does not provide the necessary finite representation. However, it has long been known that ideals of polynomials admit a finite basis [16], at least for polynomials whose coefficients are drawn from a field. This classical result, which can be interpreted as a statement on representation, adapts naturally to the setting of polynomials over modular integers, as will be explained in Sect. 3.

Example 2. Returning to Example 1, $\uparrow_{\boldsymbol{x}} Q_1$ and $\uparrow_{\boldsymbol{x}} Q_2$ admit the finite representations $\uparrow_{\boldsymbol{x}} Q_1 = \langle Q_1'\rangle_{\boldsymbol{x}}$ and $\uparrow_{\boldsymbol{x}} Q_2 = \langle Q_2\rangle_{\boldsymbol{x}}$ where $Q_1' = \{x^2+2x+1, 4x+132, y+228\}$. Observe $31^2 + 2 \cdot 31 + 1 = 1024 \equiv_{256} 0$. Similarly it follows $\gamma_{\boldsymbol{x}}(x^2+2x+1) \supseteq \{(31, y), (95, y), (159, y), (233, y) \mid y \in \mathbb{Z}_{256}\}$. Thus $x^2 + 2x + 1 \in \uparrow_{\boldsymbol{x}} Q_1$. However, $x^2 + 2x + 1 \notin \langle Q_1\rangle_{\boldsymbol{x}}$. To see this, consider the expansion of the polynomial $p(4x+132)+q(y+228) = 4(xp+33p+57q)+yq$. Observe that any term t occurring in this polynomial that is independent of y must be a term of $4(xp + 33p + 57q)$. But then, the coefficient of t must be a multiple of 4. In particular, there cannot exist p, q for which $x^2 + 2x + 1 = p(4x + 132) + q(y + 228)$, since x^2 (and in fact $2x$ and 1 as well) is independent of y but has coefficient 1. Hence Q_1 must be enlarged to obtain a basis for $\uparrow_{\boldsymbol{x}} Q_1$.

2.4 MPAD

The closure operator characterises the elements of our abstract domain:

Definition 4. $\mathit{MPAD}_m[\boldsymbol{x}] = \{P \subseteq \mathbb{Z}_m[\boldsymbol{x}] \mid \uparrow_{\boldsymbol{x}} P = P\}$

Elements of $\mathsf{MPAD}_m[\boldsymbol{x}]$ are said to be closed. If $P_1 \subseteq P_2$ then $\gamma_{\boldsymbol{x}}(P_1) \supseteq \gamma_{\boldsymbol{x}}(P_2)$ thus to align with $\langle \wp(\mathbb{Z}_m^d), \subseteq\rangle$ the domain $\mathsf{MPAD}_m[\boldsymbol{x}]$ adopts superset ordering:

Proposition 2. $\langle \mathit{MPAD}_m[\boldsymbol{x}], \sqsubseteq, \bot, \top, \sqcap, \sqcup\rangle$ *is a finite lattice, where*

$$\sqsubseteq = \supseteq \quad \bot = \mathbb{Z}_m[\boldsymbol{x}] \quad \top = \uparrow_{\boldsymbol{x}} \emptyset \quad P_1 \sqcap P_2 = \uparrow_{\boldsymbol{x}} (P_1 \cup P_2) \quad P_1 \sqcup P_2 = P_1 \cap P_2$$

Join and meet are specified set theoretically rather than algorithmically. Observe too that MPAD is finite even though there are no bounds, a priori, put on the degree of any polynomial. This follows from the finiteness of $\mathbb{Z}_m$ and the closure construction that underlies MPAD. To observe this, consider the function space $F = \{[\![p]\!]_{\boldsymbol{x}} \mid p \in \mathbb{Z}_m[\boldsymbol{x}]\} \subseteq \mathbb{Z}_m^d \to \mathbb{Z}_m$. Since the space $\mathbb{Z}_m^d \to \mathbb{Z}_m$ is finite there exists $p_1, \ldots, p_\ell \in \mathbb{Z}_m[\boldsymbol{x}]$ such that $F = \{[\![p_i]\!]_{\boldsymbol{x}} \mid i \in [1,\ell]\}$. To see how F determines the structure of $\mathsf{MPAD}_m[\boldsymbol{x}]$, define $p \equiv q$ iff $[\![q]\!]_{\boldsymbol{x}}(\boldsymbol{a}) = [\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a})$ for all $\boldsymbol{a} \in \mathbb{Z}_m^d$. Let $P \in \mathsf{MPAD}_m[\boldsymbol{x}]$ and $p \in P$. Observe $p \equiv p_i$ for some $i \in [1,\ell]$ and $\gamma_{\boldsymbol{x}}(P) \subseteq \gamma_{\boldsymbol{x}}(p) = \gamma_{\boldsymbol{x}}(p_i)$ hence $p_i \in P$. Conversely, if $p_j \in P$ and $p_j \equiv q$ then $q \in P$. Therefore there exists $I \subseteq [1,\ell]$ such that $P = \{q \in \mathbb{Z}_m[\boldsymbol{x}] \mid q \equiv p_i, i \in I\}$. Thus $\mathsf{MPAD}_m[\boldsymbol{x}]$ only has a finite number of elements.

Example 3. Continuing from Example 2, let

$$Q' = \left\{\begin{array}{l} x^3 + x + 13y^2 + 11y + 126, \\ x^2y + xy + 14y^2 + 24y, \\ 2x^2 + xy + 19y^2 + 97y + 78, \\ xy^2 + 22y^2 + 116y, \quad y^3 + 22y^2 + 72y \\ 2xy + 19y^2 + 110y, \quad 128y, \\ 4x + 2y^2 + 82y + 108, \quad 32y^2 + 64y \end{array}\right\} \quad Q = \left\{\begin{array}{l} x^2y + xy + 14y^2 + 24y, \\ xy^2 + 22y^2 + 116y, \\ 2xy + 19y^2 + 110y, \\ 4x + 2y^2 + 82y + 108, \\ y^3 + 22y^2 + 72y, \\ 32y^2 + 64y, \quad 128y \end{array}\right\}$$

Then, $\langle Q'_1\rangle_{\boldsymbol{x}} \cap \langle Q_2\rangle_{\boldsymbol{x}} = \langle Q'\rangle_{\boldsymbol{x}}$ and $\langle Q_1\rangle_{\boldsymbol{x}} \cap \langle Q_2\rangle_{\boldsymbol{x}} = \langle Q\rangle_{\boldsymbol{x}}$. Again, we defer the discussion of how Q and Q' are calculated. Observe from Figs. 1(a), 1(b) and 1(c) that $\gamma_{\boldsymbol{x}}(\langle Q'_1\rangle_{\boldsymbol{x}}) \cup \gamma_{\boldsymbol{x}}(\langle Q_2\rangle_{\boldsymbol{x}}) \subseteq \gamma_{\boldsymbol{x}}(\langle Q'\rangle_{\boldsymbol{x}})$ as required, the diamond points indicating those introduced by join itself. The diamonds in Fig. 1(d) are extraneous points introduced by calculating $\langle Q_1\rangle_{\boldsymbol{x}} \cap \langle Q_2\rangle_{\boldsymbol{x}}$ rather than $\langle Q'_1\rangle_{\boldsymbol{x}} \cap \langle Q_2\rangle_{\boldsymbol{x}}$. This illustrates that operating on arbitrary bases is not generally sufficient to maintain precision, thus motivating the need for closure.

Finally, the following result asserts that MPAD enjoys mathematical properties that simplify the application of abstract interpretation:

Proposition 3. $\langle \wp(\mathbb{Z}_m^d), \subseteq\rangle \underset{\gamma_{\boldsymbol{x}}}{\overset{\alpha_{\boldsymbol{x}}}{\rightleftharpoons}} \langle \mathit{MPAD}_m[\boldsymbol{x}], \sqsubseteq\rangle$ *is a Galois insertion, where* $\alpha_{\boldsymbol{x}}(A) = \{p \in \mathbb{Z}_m[\boldsymbol{x}] \mid A \subseteq \gamma_{\boldsymbol{x}}(p)\}$

2.5 Null Polynomials

Recall $\top = \uparrow_{\boldsymbol{x}} \emptyset = \{p \in \mathbb{Z}_m[\boldsymbol{x}] \mid \gamma_{\boldsymbol{x}}(\emptyset) \subseteq \gamma_{\boldsymbol{x}}(p)\}$. It follows $\top = \{p \in \mathbb{Z}_m[\boldsymbol{x}] \mid \forall \boldsymbol{a} \in \mathbb{Z}_m^d . [\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a}) = 0\}$ because $\gamma_{\boldsymbol{x}}(\emptyset) = \mathbb{Z}_m^d$. Such polynomials are referred to as vanishing or null polynomials [14] and represent universally valid constraints.

Example 4. Let $\boldsymbol{x} = \langle x, y\rangle$. Then in $\mathbb{Z}_{16}[\boldsymbol{x}]$, $\top = \langle N\rangle_{\boldsymbol{x}}$ where

$$N = \left\{\begin{array}{c} x^6 + x^5 + x^4 + 7x^3 + 6x^2 \ (p_1), \qquad 2x^4 + 4x^3 + 6x^2 + 4x \ (p_2), \\ x^4y^2 + x^4y + 2x^3y^2 + 2x^3y + 3x^2y^2 + 3x^2y + 2xy^2 + 2xy \ (p_3), \\ x^2y^4 + 2x^2y^3 + 3x^2y^2 + 2x^2y + xy^4 + 2xy^3 + 3xy^2 + 2xy \ (p_4), \\ y^6 + y^5 + y^4 + 7y^3 + 6y^2 \ (p_5) \qquad 2y^4 + 4y^3 + 6y^2 + 4y \ (p_6), \\ 4x^2y^2 + 4x^2y + 4xy^2 + 4xy, \quad 8x^2 + 8x, \qquad 8y^2 + 8y \end{array}\right\}$$

The p_i annotations are for future reference (Example 22).

Somewhat surprisingly an algorithm exists for finitely enumerating the set of null polynomials, hence computing $\top$, for any given number of variables and bit-width [14, Theorem 3.3]. It is tempting to remove null polynomials from bases, since they are vacuous as constraints. Unfortunately, this is not generally possible without sacrificing the canonical representation property of closure.

3 Gröbner Bases

This section provides a primer on Gröbner bases over modulo integers.

3.1 Rank and Divisibility in $\mathbb{Z}_m$

Let $| \subseteq \mathbb{Z}^2$ denote the divisibility relation over integers: $a \,|\, b$ iff b is divisible by a. The rank [26] of $a \in \mathbb{Z}_m$ is defined: $\mathsf{rank}_\omega(a) = \max\{j \in \mathbb{N} \mid 2^j \mid a\}$ if $a > 0$ otherwise ω, and can be computed by counting the number of trailing zeros in the binary representation of a [34].

Example 5. In $\mathbb{Z}_{256}$ where $\omega = 8$, $\mathsf{rank}_8(0) = 8$, $\mathsf{rank}_8(15) = 0$ and $\mathsf{rank}_8(56) = 3$.

If $a \in \mathbb{Z}_m$ then $a = 2^{\mathsf{rank}_\omega(a)}d$ for some odd d. If $a \neq 0$ then $d = a/2^{\mathsf{rank}_\omega(a)}$ is unique and the expression $2^{\mathsf{rank}_\omega(a)}d$ is referred to as the rank decomposition of a. For completeness, we declare $0 = 2^\omega \cdot 1$ be the rank decomposition of 0.

Example 6. In $\mathbb{Z}_{256}$, $0 = 2^8 \cdot 1$, $15 = 2^0 \cdot 15$ and $56 = 2^3 \cdot 7$ are rank decompositions.

For $a_1 \in \mathbb{Z}_m$ and $a_2 \in \mathbb{Z}_m \setminus \{0\}$, a_1 is divisible by a_2 if $a_1 = ba_2$ for some divisor $b \in \mathbb{Z}_m$. This occurs iff $\mathsf{rank}_\omega(a_1) \geq \mathsf{rank}_\omega(a_2)$, in which case, if $a_i = 2^{k_i}d_i$ is the rank decomposition of each a_i, then $b = 2^{k_1 - k_2}d_1{d_2}^{-1}$ where ${d_2}^{-1}$ is the multiplicative inverse of d_2 (which exists since d_2 is odd).

3.2 Monomial Orderings

Gröbner bases are founded on the concept of reduction, which simplifies a polynomial with respect to a set of polynomials. To define reduction it is necessary to order the terms in a polynomial, leading to the concept of monomial ordering:

Definition 5. *A total order $\prec$ over monomials $\boldsymbol{x}^{\boldsymbol{\alpha}}$ is a monomial ordering if: (1) $1 \prec \boldsymbol{x}^{\boldsymbol{\alpha}}$ for all $\boldsymbol{\alpha} > \mathbf{0}$ and (2) if $\boldsymbol{x}^{\boldsymbol{\alpha}_1} \prec \boldsymbol{x}^{\boldsymbol{\alpha}_2}$ then $\boldsymbol{x}^{\boldsymbol{\alpha}_1}\boldsymbol{x}^{\boldsymbol{\beta}} \prec \boldsymbol{x}^{\boldsymbol{\alpha}_2}\boldsymbol{x}^{\boldsymbol{\beta}}$ for all $\boldsymbol{x}^{\boldsymbol{\alpha}_1}$, $\boldsymbol{x}^{\boldsymbol{\alpha}_2}$ and $\boldsymbol{x}^{\boldsymbol{\beta}}$.*

If $\prec$ is a monomial ordering then $\preceq$ will denote its non-strict version. Note that monomial orderings are well-orderings, hence there is no infinite decreasing chain $\boldsymbol{x}^{\boldsymbol{\alpha}_1} \succ \boldsymbol{x}^{\boldsymbol{\alpha}_2} \succ \cdots$ of monomials.

Example 7. Let $\boldsymbol{y} = \langle x_{j_1}, \ldots, x_{j_d} \rangle$ be a permutation of $\boldsymbol{x}$ and $<$ denote lexicographical ordering over $\mathbb{N}^d$. Then, the lexicographical ordering $\prec_{\boldsymbol{y}}$, defined by $\boldsymbol{x}^{\boldsymbol{\alpha}} \prec_{\boldsymbol{y}} \boldsymbol{x}^{\boldsymbol{\beta}}$ iff $\langle \alpha_{j_1}, \ldots, \alpha_{j_d} \rangle < \langle \beta_{j_1}, \ldots, \beta_{j_d} \rangle$, is a monomial ordering.

Monomial orderings add structure to polynomials: specifically, if $p \neq 0$ then p can be uniquely expressed as $p = c\boldsymbol{x}^{\boldsymbol{\alpha}} + q$ where $c \neq 0$ and all monomials $\boldsymbol{x}^{\boldsymbol{\beta}}$ in q satisfy $\boldsymbol{x}^{\boldsymbol{\beta}} \prec \boldsymbol{x}^{\boldsymbol{\alpha}}$. Making use of this additional structure we define:

Definition 6. *Let $\prec$ be a monomial ordering over $\boldsymbol{x}$ and $p = c\boldsymbol{x}^{\boldsymbol{\alpha}} + q$ where $c \neq 0$ and all monomials $\boldsymbol{x}^{\boldsymbol{\beta}}$ in q satisfy $\boldsymbol{x}^{\boldsymbol{\beta}} \prec \boldsymbol{x}^{\boldsymbol{\alpha}}$. Then, (1) $\mathsf{lt}_{\prec}(p) = c\boldsymbol{x}^{\boldsymbol{\alpha}}$, (2) $\mathsf{lm}_{\prec}(p) = \boldsymbol{x}^{\boldsymbol{\alpha}}$ and (3) $\mathsf{lc}_{\prec}(p) = c$ are respectively the leading term, monomial and coefficient of p with respect to $\prec$.*

3.3 Reduction

Reduction is analogous to integer division with remainder:

Definition 7. *Let $p, q, r \in \mathbb{Z}_m[\boldsymbol{x}]$, $p \neq 0$, $q \neq 0$ and $\prec$ a monomial ordering. Then, p is $\prec$-reducible by q to r, denoted $p \rightarrow_{\prec,q} r$, if $\mathsf{lt}_{\prec}(p) = t\,\mathsf{lt}_{\prec}(q)$ and $p = tq + r$ for some term t.*

Reducibility lifts to sets $B \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ by $\rightarrow_{\prec,B} = \bigcup_{p \in B} \rightarrow_{\prec,p}$. Furthermore, let $\rightarrow^{+}_{\prec,B}$ (resp. $\rightarrow^{*}_{\prec,B}$) denote the transitive (resp. transitive, reflexive) closure of $\rightarrow_{\prec,B}$. If $p \rightarrow^{+}_{\prec,B} r$ for some r then p is said to be $\prec$-reducible by B, otherwise $\prec$-irreducible by B, denoted $p \not\rightarrow_{\prec,B}$.

Example 8. Let $\boldsymbol{x} = \langle x, y, a \rangle$ and $B \subseteq \mathbb{Z}_{16}[\boldsymbol{x}]$ where

$$B = \begin{Bmatrix} x + a^2 + 7a + 7 \;\; (p_1),\; y + a^2 + 7a + 7 \;\; (p_2), & \\ a^3 + a^2 + 7a + 7 \;\; (p_3),\; 2a^2 + 14 & (p_4),\; 8a + 8 \;\; (p_5) \end{Bmatrix}$$

Now, let $p = xa + 15 \in \mathbb{Z}_{16}[\boldsymbol{x}]$ and $\prec = \prec_{\boldsymbol{x}}$. Then, $\mathsf{lt}_{\prec}(p) = xa = a\,\mathsf{lt}_{\prec}(p_1)$ and $p = ap_1 + r_1$ where $r_1 = 15a^3 + 9a^2 + 9a + 15$, hence $p \rightarrow_{\prec,p_1} r_1$. Similarly, $\mathsf{lt}_{\prec}(r_1) = 15a^3 = 15\,\mathsf{lt}_{\prec}(p_3)$ and $r_1 = 15p_3 + r_2$ where $r_2 = 10a^2 + 6$, hence $r_1 \rightarrow_{\prec,p_3} r_2$. Finally, $\mathsf{lt}_{\prec}(r_2) = 10a^2 = 5\,\mathsf{lt}_{\prec}(p_4)$ and $r_2 = 5p_4 + r_3$ where $r_3 = 0$, hence $r_2 \rightarrow_{\prec,p_4} r_3$. Thus, $p \rightarrow_{\prec,p_1} r_1 \rightarrow_{\prec,p_3} r_2 \rightarrow_{\prec,p_4} r_3$, hence $p \rightarrow^{+}_{\prec,B} 0$.

Note p is $\prec$-reducible by B iff $\mathsf{lt}_{\prec}(p)$ is divisible by $\mathsf{lt}_{\prec}(q)$ for some $q \in B$, where a term t_1 is divisible by a term t_2 if $t_1 = t_2 t_3$ for some term t_3. Moreover, reduction eliminates the leading term of a polynomial, leaving a residue polynomial comprised of strictly smaller terms with respect to $\prec$:

Lemma 2. *If $p \rightarrow^{+}_{\prec,B} r \neq 0$ then $\mathsf{lm}_{\prec}(r) \prec \mathsf{lm}_{\prec}(p)$.*

Since monomial orderings are well-orderings, the previous result implies that a sequence of reductions cannot continue ad infinitum and must eventually terminate with the 0 polynomial. In this case, it follows that $p \in \langle B \rangle_{\boldsymbol{x}}$, hence reduction provides a test for membership in an ideal:

Proposition 4. *If $p \rightarrow^{*}_{\prec,B} 0$ then $p \in \langle B \rangle_{\boldsymbol{x}}$.*

But reduction against an arbitrary basis B does not lead to a complete test for membership in $\langle B \rangle_{\boldsymbol{x}}$, hence motivating Gröbner bases.

3.4 Gröbner Bases

With reduction in place, the concept of Gröbner basis can be introduced:

Definition 8. *Let $B \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ and $\prec$ a monomial ordering over $\boldsymbol{x}$. Then, $G \subseteq \langle B\rangle_{\boldsymbol{x}}$ is a Gröbner basis for $\langle B\rangle_{\boldsymbol{x}}$ with respect to $\prec$ if for all $p \in \langle B\rangle_{\boldsymbol{x}}$, if $p \neq 0$ then p is $\prec$-reducible by G.*

Gröbner bases provide a complete test for membership in $\langle B\rangle_{\boldsymbol{x}}$, as asserted by:

Lemma 3. *If G is a Gröbner basis for $\langle B\rangle_{\boldsymbol{x}}$ with respect to $\prec$ then for all $p \in \langle B\rangle_{\boldsymbol{x}}$, $p \rightarrow^*_{\prec,G} 0$.*

Example 9. Let $\boldsymbol{x} = \langle x, y\rangle$, $\prec = \prec_{\boldsymbol{x}}$ and $p \in \mathbb{Z}_{16}[\boldsymbol{x}]$ where $p = 4x$. Moreover, let $B = \{p_1, p_2\} \subseteq \mathbb{Z}_{16}[\boldsymbol{x}]$ where $p_1 = 2x^2y + 2x^2 + 6xy + x$ and $p_2 = 4y + 4$. Then, $p = 12p_1 + (10x^2 + 10x)p_2 \in \langle B\rangle_{\boldsymbol{x}}$, yet $p \not\rightarrow_{\prec,B}$, thus B is not a Gröbner basis with respect to $\prec$. However, it can be shown if $p_3 = 6x$ and $p_4 = 3x$ then $G = \{p_1, p_2, p_3, p_4\}$ is a Gröbner basis for $\langle B\rangle_{\boldsymbol{x}}$ with respect to $\prec$. Note that p is $\prec$-reducible by $p_4 \in G$. Indeed, $p \rightarrow_{\prec,p_4} 0$, hence $p \rightarrow^*_{\prec,G} 0$, as predicted by the previous result.

3.5 Buchberger's Algorithm

Classically [3], Gröbner bases are computed by evaluating S-polynomials:

Definition 9. *Let $\prec$ be a monomial ordering over $\boldsymbol{x}$. The S-polynomial of $p_1, p_2 \in \mathbb{Z}_m[\boldsymbol{x}]$ with respect to $\prec$ is defined:*

$$\mathsf{S}_{\prec}(p_1, p_2) = d_2 2^{k-k_1} \boldsymbol{x}^{\boldsymbol{\alpha}-\boldsymbol{\alpha}_1} p_1 - d_1 2^{k-k_2} \boldsymbol{x}^{\boldsymbol{\alpha}-\boldsymbol{\alpha}_2} p_2$$

where, if $p_i = 0$ then $k_i = \omega$, $d_i = 1$ and $\boldsymbol{\alpha}_i = \mathbf{0}$, else $2^{k_i} d_i$ is the rank decomposition of $\mathsf{lc}_{\prec}(p_i)$ and $\boldsymbol{x}^{\boldsymbol{\alpha}_i} = \mathsf{lm}_{\prec}(p_i)$, $k = \max(k_1, k_2)$ and $\boldsymbol{\alpha} = \max(\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2})$.

Example 10. If $p_1 = 2x^2y + 2x^2 + 6xy + x$ and $p_2 = 4y + 4$ then it follows $\mathsf{S}_{\prec}(p_1, p_2) = 2(2xy^2 + 6xy + 2y^2 + y) - y^2(4x + 4) = 12xy + 2y$ and $\mathsf{S}_{\prec}(p_1, 0) = 8(2xy^2 + 6xy + 2y^2 + y) - xy^2(0) = 8y$.

Note that $\mathsf{lt}_{\prec}(d_2 2^{k-k_1} \boldsymbol{x}^{\boldsymbol{\alpha}-\boldsymbol{\alpha}_1} p_1) = \mathsf{lt}_{\prec}(d_1 2^{k-k_2} \boldsymbol{x}^{\boldsymbol{\alpha}-\boldsymbol{\alpha}_2} p_2)$, hence the S-polynomial $\mathsf{S}_{\prec}(p_1, p_2)$ leads to a cancellation of leading terms. In particular, the S-polynomial $\mathsf{S}_{\prec}(p_1, 0)$ eliminates the leading term of p_1, and possible other terms as well. This deviates from the classical case of fields, where only multiplying by 0 can eliminate a leading term. S-polynomials then yield an effective criterion [2, Theorem 30] to determine if a given basis is a Gröbner basis.

Theorem 1 (Buchberger's criterion). *Let $\prec$ be a monomial ordering and $B = \{p_1, \ldots, p_s\} \subseteq \mathbb{Z}_m[\boldsymbol{x}]$. If $\mathsf{S}_{\prec}(p_i, p_j) \rightarrow^*_{\prec,B} 0$ and $\mathsf{S}_{\prec}(p_i, 0) \rightarrow^*_{\prec,B} 0$ for all $1 \leq i < j \leq s$ then B is a Gröbner basis for $\langle B\rangle_{\boldsymbol{x}}$ with respect to $\prec$.*

```
function gb_≺(B = {p_1, ..., p_s} ⊆ ℤ_m[x])
begin
    G := B
    S := {(p_i, p_j) | 1 ≤ i < j ≤ s} ∪ {(p_i, 0) | 1 ≤ i ≤ s}
    while (S ≠ ∅)
        let s = (f_1, f_2) ∈ S
        S := S \ {s}
        p := S_≺(f_1, f_2)
        let p →*_{≺,G} r where r ↛_{≺,G}
        if (r ≠ 0)
            S := S ∪ {(g, r) | g ∈ G} ∪ {(r, 0)}
            G := G ∪ {r}
        end if
    end while
    return G
end
```

Fig. 2. Gröbner basis algorithm over integers modulo 2^ω

Buchberger's criterion justifies Buchberger's algorithm for constructing Gröbner bases. Figure 2 presents a version of Buchberger's algorithm [2] that takes $B \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ and a monomial ordering $\prec$ over $\boldsymbol{x}$ and returns a Gröbner basis for $\langle B\rangle_{\boldsymbol{x}}$ with respect to $\prec$. The algorithm maintains a basis G, initialised to B, and a set of unverified S-polynomials S. The algorithm attempts to verify that G is a Gröbner basis by reducing each S-polynomial pair in S against it. If some S-polynomial does not reduce, it yields a new element which is added to G, and generates further S-polynomials. The algorithm terminates when all S-polynomials for the current basis reduce to 0, at which point Buchberger's criterion applies to show the result, henceforth denoted $\mathsf{gb}_\prec(B)$, is a Gröbner basis. Observe that $B \subseteq G$ on each iteration of the while loop hence $B \subseteq \mathsf{gb}_\prec(B)$.

4 Calculating Variable Elimination and Join

This section explains how variable elimination can be computed using Gröbner bases, and how variable elimination can be combined with a relaxation to compute the join of two ideals finitely represented as bases.

4.1 Variable Elimination

A generic projection function $\pi_i(\langle a_1, \ldots, a_\ell\rangle) = \langle a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_\ell\rangle$ is used to formulate elimination. The presentation of elimination commences with a syntactic version which removes polynomials that contains a given variable:

Definition 10. *(Syntactic) variable elimination is an operation* $\mathsf{elim}[x_j]$ *where* $\mathsf{elim}[x_j] : \wp(\mathbb{Z}_m[\boldsymbol{x}]) \to \wp(\mathbb{Z}_m[\pi_j(\boldsymbol{x})])$ *defined by* $\mathsf{elim}[x_j](P) = P \cap \mathbb{Z}_m[\pi_j(\boldsymbol{x})]$

The following result demonstrates that abstraction and elimination commute. The result is formulated in terms of the natural lifting of π_j from the function space $\mathbb{Z}_m^d \to \mathbb{Z}_m^{d-1}$ to $\wp(\mathbb{Z}_m^d) \to \wp(\mathbb{Z}_m^{d-1})$.

Proposition 5. *If $A \subseteq \mathbb{Z}_m^d$ then $\mathsf{elim}[x_j](\alpha_{\boldsymbol{x}}(A)) = \alpha_{\pi_j(\boldsymbol{x})}(\pi_j(A))$.*

It follows from this result that elimination preserves closure:

Corollary 1. *If $P \in \mathsf{MPAD}_m[\boldsymbol{x}]$ then $\mathsf{elim}[x_j](P) \in \mathsf{MPAD}_m[\pi_j(\boldsymbol{x})]$.*

Example 11. Consider $B = \{wx + 10w, 15wx^2 + wx + x^2 + 15x\} \subseteq \mathbb{Z}_{16}[w, x]$ and observe $\mathsf{elim}[w](B) = \emptyset$. However $(x^2 + 7x + 8)(wx + 10w) + (x + 2)(15wx^2 + wx + x^2 + 15x) = x^3 + x^2 + 14x$ hence $x^3 + x^2 + 14x \in \langle B\rangle_{\langle w,x\rangle}$. Since $w \notin \mathsf{vars}(x^3 + x^2 + 14x)$ it follows $x^3 + x^2 + 14x \in \mathsf{elim}[w](\langle B\rangle_{\langle w,x\rangle})$. In particular, $\mathsf{elim}[w](\langle B\rangle_{\langle w,x\rangle}) \neq \{0\} = \langle\emptyset\rangle_{\langle x\rangle} = \langle\mathsf{elim}[w](B)\rangle_{\langle x\rangle}$.

The previous example shows that syntactic variable elimination is not well-behaved with respect to ideal generation, thus motivating the following:

Definition 11. *(Semantic) variable elimination is a relation $\to_{\mathsf{elim}[x_j]}$ where $\to_{\mathsf{elim}[x_j]} \subseteq \wp(\mathbb{Z}_m[\boldsymbol{x}]) \times \wp(\mathbb{Z}_m[\pi_j(\boldsymbol{x})])$ defined by $B \to_{\mathsf{elim}[x_j]} B'$ iff $\mathsf{elim}[x_j](\langle B\rangle_{\boldsymbol{x}}) = \langle B'\rangle_{\pi_j(\boldsymbol{x})}$*

Proposition 6. *Let $B \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ and B' be a Gröbner basis for $\langle B\rangle_{\boldsymbol{x}}$ with respect to $\prec_{\boldsymbol{y}}$ where $\boldsymbol{y}$ is a permutation of $\boldsymbol{x}$ and $y_1 = x_j$. Then $B \to_{\mathsf{elim}[x_j]} \mathsf{elim}[x_j](B')$.*

The previous result can be stated more generally in terms of elimination orderings [1]; the restriction to lexicographical ordering is adopted merely to simplify the presentation. Consistent with this choice, $\mathsf{gb}_{\prec_{\boldsymbol{y}}}$ is henceforth abbreviated to $\mathsf{gb}_{\boldsymbol{y}}$, again purely to streamline the exposition.

Example 12. Let $B = \{wx + 10w, 15wx^2 + wx + x^2 + 15x\} \subseteq \mathbb{Z}_{16}[w, x, y]$. Then, $\mathsf{gb}_{\langle w,x,y\rangle}(B) = B \cup \{wx + 3x^2 + 13x, 2w + x^2 + 15x, x^3 + x^2 + 14x\}$. It therefore follows $B \to_{\mathsf{elim}[w]} \{x^3 + x^2 + 14x\}$.

Example 13. Let $B = \{w(x+3), w(y+9), (1-w)(x+6), (1-w)(y+2)\}$. Then,

$$\begin{aligned}\mathsf{gb}_{\langle w,x,y\rangle}(B) &= B \cup \{w + 7y + 14, \quad x + 5y, \quad y^2 + 11y + 2\}\\ \mathsf{gb}_{\langle w,y,x\rangle}(B) &= B \cup \{w + 5x + 14, \quad y + 13x, \quad x^2 + 9x + 2\}\end{aligned}$$

Thus $B \to_{\mathsf{elim}[w]} B'$ and $B \to_{\mathsf{elim}[w]} B''$ where $B' = \{x + 5y, y^2 + 11y + 2\}$ and $B'' = \{y + 13x, x^2 + 9x + 2\}$ illustrating why $\to_{\mathsf{elim}[w]}$ is defined as a relation. To see $\langle B'\rangle_{\langle x,y\rangle} = \langle B''\rangle_{\langle x,y\rangle}$ observe $x + 5y \to_{y+13x} 0$ and

$$\begin{aligned}y^2 + 11y + 2 &\to_{y+13x} 3xy + 11y + 2\\ &\to_{y+13x} 9x^2 + 11y + 2 \to_{x^2+9x+2} 15x + 11y \to_{y+13x} 0\end{aligned}$$

Similarly, $p \to_{B'} 0$ for all $p \in B''$.

4.2 Join

Once variable elimination is in place, join can be calculated by adapting a standard relaxation [1] to the current setting. The result, which provides a way of intersecting ideals, hence calculating join, is stated in terms of a lifted product $qP = \{qp \mid p \in P\}$ where $P \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ and $q \in \mathbb{Z}_m[\boldsymbol{x}]$:

Proposition 7. *Let* $\langle B_1\rangle_{\boldsymbol{x}}, \langle B_2\rangle_{\boldsymbol{x}} \in \mathit{MPAD}_m[\boldsymbol{x}]$. *If* $w \notin \mathsf{vars}(B_1 \cup B_2)$ *then* $\langle B_1\rangle_{\boldsymbol{x}} \cap \langle B_2\rangle_{\boldsymbol{x}} = \langle B\rangle_{\boldsymbol{x}}$ *whenever* $wB_1 \cup (1-w)B_2 \rightarrow_{\mathit{elim}[w]} B$

Example 14. Let $\boldsymbol{x} = \langle x, y\rangle$ and $B_1, B_2 \subseteq \mathbb{Z}_{16}[\boldsymbol{x}]$ where $B_1 = \{x + 10\}$, $B_2 = \{x^2 + 15x\}$ and $I_i = \langle B_i\rangle_{\boldsymbol{x}}$. Both I_i are closed, that is, $I_i = \uparrow I_i$. Let

$$B = wB_1 \cup (1-w)B_2 = \{wx + 10w, 15wx^2 + wx + x^2 + 15x\}$$

By Example 11, $B \rightarrow_{\mathsf{elim}[w]} \{x^3 + x^2 + 14x\}$, hence $\langle B_1\rangle_{\boldsymbol{x}} \sqcup \langle B_2\rangle_{\boldsymbol{x}} = \langle x^3 + x^2 + 14x\rangle_{\boldsymbol{x}}$.

Figures 3(a), 3(b) and 3(i) depict $\gamma_{\boldsymbol{x}}(I_1)$, $\gamma_{\boldsymbol{x}}(I_2)$ and $\gamma_{\boldsymbol{x}}(I_1 \sqcup I_2)$ respectively. Observe $(8, y) \in \gamma_{\boldsymbol{x}}(I_1 \sqcup I_2)$ but $(8, y) \notin \gamma_{\boldsymbol{x}}(I_1) \cup \gamma_{\boldsymbol{x}}(I_2)$ for any $y \in \mathbb{Z}_{16}$. These additional points, which are introduced by join itself, stem not from the relaxation $wB_1 \cup (1-w)B_2$ which introduces w, but the elimination of w from $\mathsf{gb}_{\langle w,x,y\rangle}(B)$ which derives a unary polynomial representation over x alone. To see this, observe $B[x \mapsto 8] = \{8w, 8w + 8\}$ and $B[x \mapsto 14] = \{14w + 12, 10w + 6\}$ both have no solutions.

Example 15. Figure 3 presents a series of examples of join on $\mathbb{Z}_{16}[\boldsymbol{x}]$ for $\boldsymbol{x} = \langle x, y\rangle$. Figures 3(a)–(h) depict $\gamma_{\boldsymbol{x}}(I_i)$ for $I_i = \langle B_i\rangle_{\boldsymbol{x}}$ where $I_i = \uparrow I_i$ and B_i are as follows:

$$\begin{array}{ll} B_3 = \{x + 3,\quad y + 9\} & B_6 = \{x^2,\quad xy^4 + xy^2 + 2xy,\quad 2xy^2 + 2xy,\quad 4x\} \\ B_4 = \{x + 6,\quad y + 2\} & B_7 = \{x^4y + x^2y + 2xy,\quad 2x^2y + 2xy,\quad y^2,\quad 4y\} \\ B_5 = \{x^2,\quad 4x,\quad y\} & B_8 = \{x + y\} \end{array}$$

For comparison, the yellow points give the best abstraction of $\gamma_{\boldsymbol{x}}(I_i)$ using systems of linear congruences modulo 16 (linear polynomials).

Figures 3(i)–(p) depict $\gamma_{\boldsymbol{x}}(I_i \sqcup I_j)$ for various combinations of $i, j \in \{1, \ldots, 8\}$, illustrating where a polynomial representation introduces additional points via join. Observe that join induces a loss of information as witnessed by additional points of the form $(8, y)$ and $(14, y)$ in $\gamma_{\boldsymbol{x}}(I_1 \sqcup I_2)$. Again, the yellow points give the join of the best linear abstractions, which can be computed by combining a relaxation with variable elimination [21]. To illustrate the working, consider B_3 and B_4 rewritten as follows:

$$B_3 = \{\, x \equiv_{16} -3,\quad y \equiv_{16} -9\,\} \qquad B_4 = \{\, x \equiv_{16} -6,\quad y \equiv_{16} -2\,\}.$$

The relaxation introduces fresh variables x', y', x'', y'' and μ:

$$\begin{array}{lll} x \equiv_{16} x' + x'' & x' \equiv_{16} -3\mu & x'' \equiv_{16} -6(1-\mu) \\ y \equiv_{16} y' + y'' & y' \equiv_{16} -9\mu & y'' \equiv_{16} -2(1-\mu) \end{array}$$

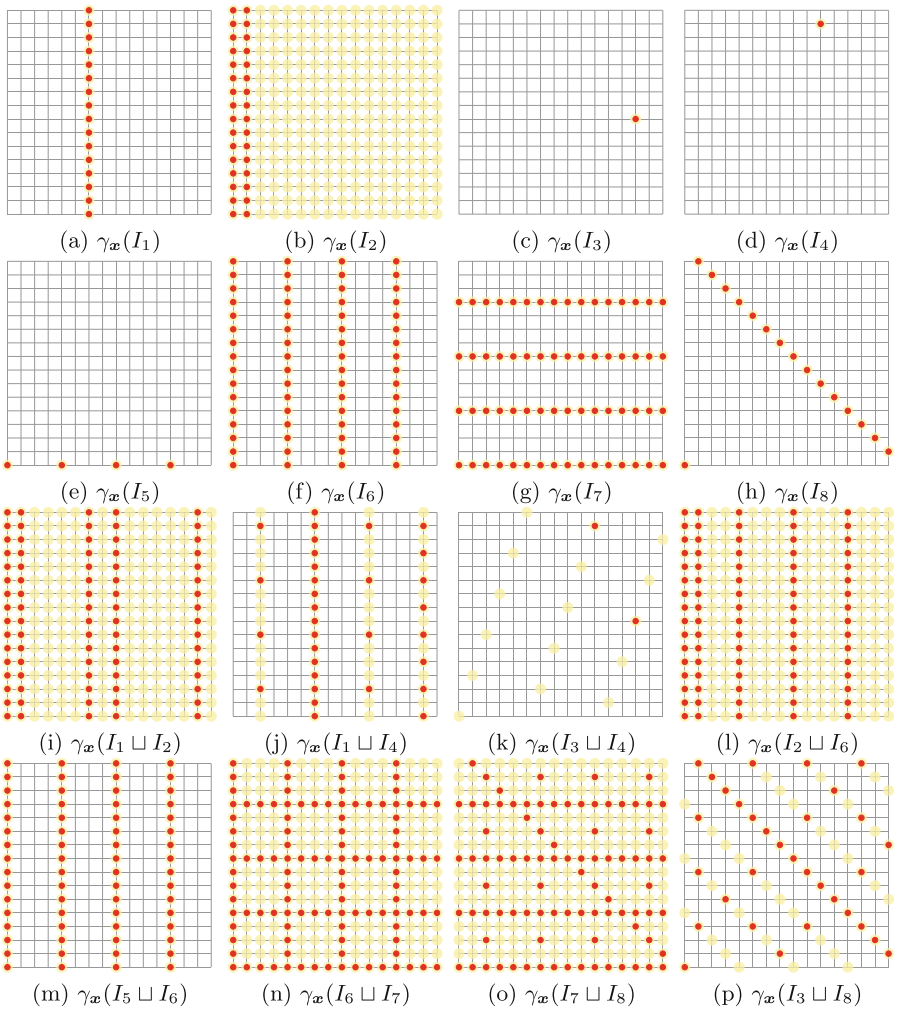


Fig. 3. Examples of join on $\mathbb{Z}_{16}[\boldsymbol{x}]$ for $\boldsymbol{x} = \langle x, y \rangle$

Eliminating x', y', x'' and y'' gives a system of two congruences: $x \equiv_{16} 3\mu - 6$ and $y \equiv_{16} -7\mu - 2$. Rearranging for μ gives $\mu \equiv_{16} 2 - 5x$ hence $y \equiv_{16} 3x$ as illustrated in Fig. 3(k). The other linear joins are computed likewise.

Note the loss of precision in using linear, rather than polynomial, abstractions. For instance, the set $\gamma_{\boldsymbol{x}}(I_2)$ can only be approximated by a trivial (unconstrained) linear system, which loses all information. Moreover, as demonstrated in Figs. 3(j)–(k) and Figs. 3(n)–(p), even if the arguments to a (polynomial) join are representable via linear systems, the result may not be.

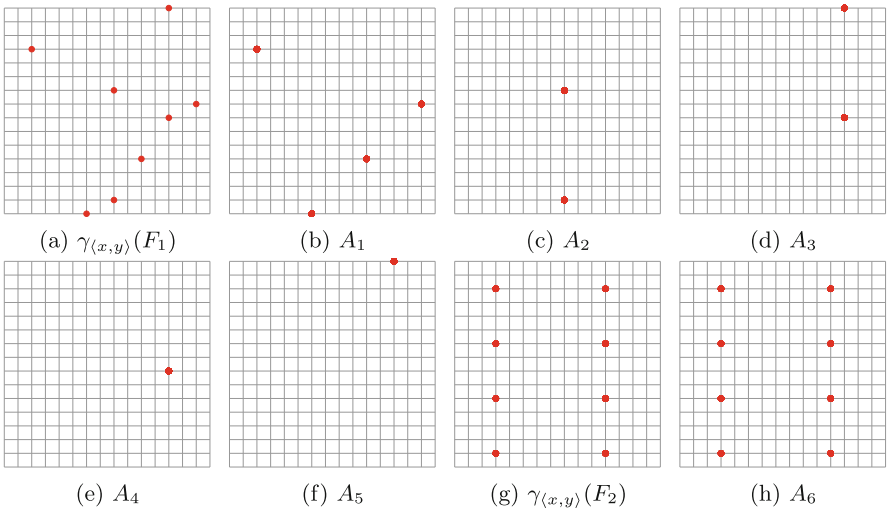


Fig. 4. Covers of F_1 over $\langle w_1\rangle$ and F_2 over $\langle w_1, w_2\rangle$

5 Calculating Closure and Meet

This section addresses how to finitely compute closure. The problem is reduced to that of computing a cover of a system of polynomials. A cover provides a way to decompose closure to sub-problems for which closure can be computed directly. A divide-and-conquer algorithm is introduced for computing a cover, which exploits a simplification procedure based on Gröbner bases, to avoid superfluous work. The section concludes by showing how meet can be computed using closure.

5.1 Covering

An algorithm for closure is formulated in terms of the concept of a cover, which is itself defined through a lifting of polynomial evaluation $[\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a})$ to a vector of polynomials $\boldsymbol{p} = \langle p_1, \ldots, p_n\rangle$ by $[\![\boldsymbol{p}]\!]_{\boldsymbol{x}}(\boldsymbol{a}) = \langle [\![p_1]\!]_{\boldsymbol{x}}(\boldsymbol{a}), \ldots, [\![p_n]\!]_{\boldsymbol{x}}(\boldsymbol{a})\rangle$.

Definition 12. *Let $\mathcal{W} \subseteq \mathbb{Z}_m[\boldsymbol{w}]^d$, $A \subseteq \mathbb{Z}_m^d$ and $F \subseteq \mathbb{Z}_m[\boldsymbol{x}]$. Then*

- *$\mathcal{W}$ is a cover of A over $\boldsymbol{w}$ iff $A = \{[\![\boldsymbol{W}]\!]_{\boldsymbol{w}}(\boldsymbol{a}) \mid \boldsymbol{W} \in \mathcal{W} \wedge \boldsymbol{a} \in \mathbb{Z}_m^{|\boldsymbol{w}|}\}$*
- *$\mathcal{W}$ is a cover of F over $\boldsymbol{w}$ iff $\mathcal{W}$ is a cover of $\gamma_{\boldsymbol{x}}(F)$ over $\boldsymbol{w}$*

Example 16. Figures 4(a) and (g) depict $\gamma_{\boldsymbol{x}}(F_1)$ and $\gamma_{\boldsymbol{x}}(F_2)$ for $\boldsymbol{x} = \langle x, y\rangle$ where

$$F_1 = \left\{x + 3y^3 + 4y^2 + 7y + 10, \quad y^4 + 7y^2 + 8y\right\} \qquad F_2 = \left\{2x + 10, \quad 4y + 12\right\}$$

Figures 4(b), (c) and (d) illustrate $A_i = \{[\![\boldsymbol{W}_i]\!]_{\boldsymbol{w}}(\boldsymbol{a}) \mid \boldsymbol{a} \in \mathbb{Z}_m^1\}$ for $\boldsymbol{w} = \langle w_1\rangle$ where $\boldsymbol{W}_1 = \langle 4w_1+6, 4w_1\rangle$, $\boldsymbol{W}_2 = \langle 8, 8w_1+1\rangle$ and $\boldsymbol{W}_3 = \langle 12, 8w_1+7\rangle$. Observe

```
function cover(F ⊆ ℤ_m[x])
begin
    let w = ⟨w_1, ..., w_d⟩
    return cover(w, F[x_1 ↦ w_1, ..., x_d ↦ w_d])
end
function cover(S ∈ ℤ_m[w]^d × ℘(ℤ_m[w]))
begin
    S' = simplify(S)
    if (S' = nil) return ∅
    else
        let ⟨W, F⟩ = S'
        if (F = ∅) return {W}
        else
            let w_i ∈ vars(F)
            S'_0 = constrain(S', 1, w_i, 0)          (* F ∪ {w_i − 2^1 w} *)
            S'_1 = constrain(S', 1, w_i, 1)          (* F ∪ {w_i − 2^1 w + 1} *)
            return cover(S'_0) ∪ cover(S'_1)
        end if
    end if
end
```

Fig. 5. The cover algorithm

$\{\boldsymbol{W}_i\}$ is a cover of A_i and since $\gamma_{\boldsymbol{x}}(F_1) = A_1 \cup A_2 \cup A_3$, $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3\}$ is a cover of F_1 over $\boldsymbol{w}$. The set of 4 vectors $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_4, \boldsymbol{W}_5\}$ where $\boldsymbol{W}_4 = \langle 12, 7\rangle$ and $\boldsymbol{W}_5 = \langle 12, 15\rangle$ is also a cover of F_1, illustrating that covers are not unique. The polynomial vectors $\boldsymbol{W}_4$ and $\boldsymbol{W}_5$ define single points and suggest how a cover can be constructed for an arbitrary $F \subseteq \mathbb{Z}_m[\boldsymbol{w}]$ by putting $\mathcal{W} = \{\boldsymbol{a} \mid \boldsymbol{a} \in \gamma_{\boldsymbol{x}}(F)\}$. The vector $\boldsymbol{w}$ is not necessarily unary as the cover $\{\boldsymbol{W}_6\}$ of F_2 over $\boldsymbol{w} = \langle w_1, w_2\rangle$ illustrates where $\boldsymbol{W}_6 = \langle 8w_1 + 3, 4w_2 + 1\rangle$ and $\gamma_{\boldsymbol{x}}(F_2) = A_6 = \{[\![\boldsymbol{W}_6]\!]_{\boldsymbol{w}}(\boldsymbol{a}) \mid \boldsymbol{a} \in \mathbb{Z}_m^2\}$, and $\gamma_{\boldsymbol{x}}(F_2)$ and A_6 are illustrated in Figs. 4(g) and (h) respectively.

The challenge is to compute a cover over some $\boldsymbol{w}$ for arbitrary $F \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ without naively enumerating all points of $\gamma_{\boldsymbol{x}}(F)$. To this end, Fig. 5 presents a divide-and-conquer algorithm that recursively decomposes $\gamma_{\boldsymbol{x}}(F)$ into subsets following the structure of F. Ultimately the function computes a cover $\mathcal{W} \subseteq \mathbb{Z}_m[\boldsymbol{w}]^d$ for F over $\boldsymbol{w}$ where $|\boldsymbol{w}| = d = |\boldsymbol{x}|$. The function cover depends on three auxiliary functions, simplify, constrain and safe all of which are listed in Fig. 6. The function cover and its auxiliaries operate on pairs $S = \langle \boldsymbol{W}, F\rangle$ where $\boldsymbol{W} \in \mathbb{Z}_m[\boldsymbol{w}]^d$ is a vector of polynomials and $F \subseteq \mathbb{Z}_m[\boldsymbol{w}]$ is a system. The vector $\boldsymbol{W}$ provides a lens to interpret the solutions of F, as is formalised below:

Definition 13. *The concretisation map* $\gamma_{\boldsymbol{w}} : \mathbb{Z}_m[\boldsymbol{w}]^d \times \wp(\mathbb{Z}_m[\boldsymbol{w}]) \to \wp(\mathbb{Z}_m^d)$ *is defined:* $\gamma_{\boldsymbol{w}}(\langle \boldsymbol{W}, F\rangle) = \{[\![\boldsymbol{W}]\!]_{\boldsymbol{w}}(\boldsymbol{a}) \mid \boldsymbol{a} \in \gamma_{\boldsymbol{w}}(F)\}$

function simplify($\langle \boldsymbol{W}, F\rangle \in \mathbb{Z}_m[\boldsymbol{w}]^d \times \wp(\mathbb{Z}_m[\boldsymbol{w}])$)
begin
  $F' = \mathsf{gb}_{\boldsymbol{w}}(F)$
  $S' = \langle \boldsymbol{W}, F'\rangle$
  if ($c \in F'$ where $c \in \mathbb{Z}_m \setminus \{0\}$)
    return nil
  else if ($2^{\omega-j}(w_i + r) \in F'$ where $j > 0 \wedge r \in \mathbb{Z}_m[w_{i+1}, \ldots, w_d] \wedge \mathsf{safe}(\boldsymbol{W}, w_i, r)$)
    $S'' = \mathsf{constrain}(S', j, w_i, r)$ (* $F \cup \{w_i - 2^j w + r\}$ *)
    return simplify(S'')
  else
    return S'
  end if
end
function constrain($\langle \boldsymbol{W}, F\rangle \in \mathbb{Z}_m[\boldsymbol{w}]^d \times \mathbb{Z}_m[\boldsymbol{w}], j \in \mathbb{N}, w_i \in \boldsymbol{w}, r \in \mathbb{Z}_m[w_{i+1}, \ldots, w_d]$)
begin
  $F \cup \{w_i - 2^j w + r\} \rightarrow_{\mathsf{elim}[w_i]} F'$
  $\boldsymbol{W}' = \boldsymbol{W}[w_i \mapsto 2^j w - r]$
  if ($W_i' = 2^\omega w + q \wedge q \in \mathbb{Z}_m[w_{i+1}, \ldots, w_d]$) $F'' = F'[w \mapsto 0]$
  else $F'' = F'[w \mapsto w_i]$
  return $\langle \boldsymbol{W}'[w \mapsto w_i], F''\rangle$
end
function safe($\boldsymbol{W} \in \mathbb{Z}_m[\boldsymbol{w}]^d, w_i \in \boldsymbol{w}, r \in \mathbb{Z}_m[w_{i+1}, \ldots, w_d]$)
begin
  let $\boldsymbol{W} = \langle 2^{k_1} w_1 + q_1, \ldots, 2^{k_d} w_d + q_d\rangle$
  if ($c\boldsymbol{y}^{\boldsymbol{\alpha}} \in r, w_\ell \in \mathsf{vars}(\boldsymbol{y})$ where $k_i + \mathsf{rank}(c) < k_\ell$) return false
  else return true
end

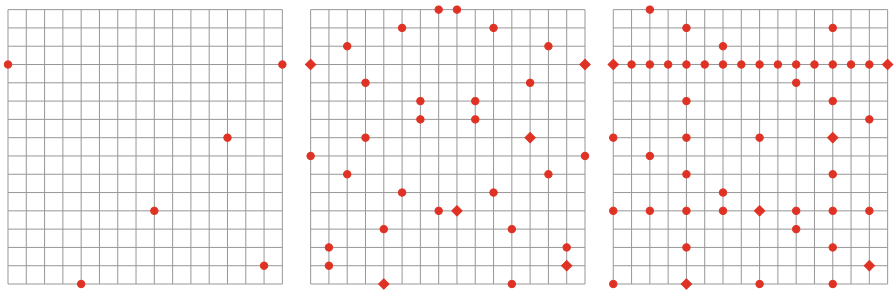
Fig. 6. The simplify, constrain and safe functions

Example 17. Consider $S_b = \langle \boldsymbol{W}_b, F_b\rangle$ and $S_c = \langle \boldsymbol{W}_c, F_c\rangle$, where $\boldsymbol{W}_b = \langle w_1, 2w_2\rangle$, $\boldsymbol{W}_c = \langle w_1, 4w_2\rangle$ and

$$F_b = \begin{Bmatrix} w_1^2 + w_1 + 6w_2 + 12, \\ 2w_1w_2 + 4w_1, \quad 4w_2^2, \quad 8w_2 \end{Bmatrix} \qquad F_c = \begin{Bmatrix} w_1^2 + w_1 + 12w_2 + 12, \\ 4w_1w_2 + 4w_1 \end{Bmatrix}$$

Figure 8(b) illustrates $\gamma_{\boldsymbol{w}}(F_b)$ as translucent points and $\gamma_{\boldsymbol{w}}(S_b)$ as opaque points. Observe $\langle 8, 2\rangle, \langle 8, 10\rangle \in \gamma_{\boldsymbol{w}}(F_b)$ and $[\![\boldsymbol{W}_b]\!]_{\boldsymbol{w}}(\langle 8, 2\rangle) = \langle 8, 4\rangle = [\![\boldsymbol{W}_b]\!]_{\boldsymbol{w}}(\langle 8, 10\rangle)$. Hence, in general, there is a many-to-one relationship between $\gamma_{\boldsymbol{w}}(F_b)$ and $\gamma_{\boldsymbol{w}}(S_b)$. Figure 8(c) depicts $\gamma_{\boldsymbol{w}}(F_c)$ and $\gamma_{\boldsymbol{w}}(S_c)$ using the same convention. Observe too that $\gamma_{\boldsymbol{w}}(S_b) = \gamma_{\boldsymbol{w}}(S_c)$ but the cardinality of $\gamma_{\boldsymbol{w}}(F_c)$ is 4-fold that of $\gamma_{\boldsymbol{w}}(S_c)$ since $\boldsymbol{W}_c = \langle w_1, 4w_2\rangle$.

Observe that if $\mathcal{W} \subseteq \mathbb{Z}_m[\boldsymbol{w}]^d$ is a cover for $F \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ over $\boldsymbol{w}$ then $\gamma_{\boldsymbol{x}}(F) = \cup\{\gamma_{\boldsymbol{w}}(\langle \boldsymbol{W}, \emptyset\rangle) \mid \boldsymbol{W} \in \mathcal{W}\}$. Thus a cover is formed from pairs $\langle \boldsymbol{W}, F\rangle$ that are degenerate in that $F = \emptyset$. The rationale behind cover is thus to decompose a single pair $\langle \boldsymbol{W}, F\rangle$ where $\boldsymbol{W} = \boldsymbol{w}$ into a collection of degenerate pairs:



(a) $\gamma_{\boldsymbol{x}}(F)$ (b) $\gamma_{\boldsymbol{x}}(F_3)$ (c) $\gamma_{\boldsymbol{x}}(F_4)$

Fig. 7. Solution sets for F, F_3 and F_4

Example 18. Consider computing a cover for the system

$$F = \left\{x^2 + x + 7y^2 + 11y + 12, \quad xy + 4x + 10y^2\right\}$$

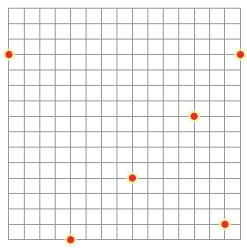
over $\boldsymbol{w} = \langle w_1, w_2\rangle$. The set $\gamma_{\boldsymbol{x}}(F)$ is plotted in Fig. 7(a). The top-level cover function expresses F as the pair $S_a = \langle \boldsymbol{W}_a, F_a\rangle$ where

$$\boldsymbol{W}_a = \boldsymbol{w} \qquad F_a = \left\{w_1^2 + w_1 + 7w_2^2 + 11w_2 + 12, \quad w_1w_2 + 4w_1 + 10w_2^2\right\}$$

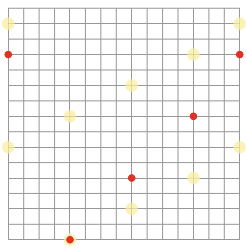
Since $[\![\boldsymbol{W}_a]\!]_{\boldsymbol{w}}(\boldsymbol{b}) = \boldsymbol{b}$ for all $\boldsymbol{b} \in \mathbb{Z}_m^2$, it follows $\gamma_{\boldsymbol{w}}(S_a) = \gamma_{\boldsymbol{x}}(F)$.

The cover function invokes both simplify and constrain. The function simplify performs simplification, either returning nil, indicating $\gamma_{\boldsymbol{w}}(\langle \boldsymbol{W}, F\rangle) = \emptyset$, or $S' = \langle \boldsymbol{W}', F'\rangle$ where $\gamma_{\boldsymbol{w}}(S) = \gamma_{\boldsymbol{w}}(S')$ (possibly with $S = S'$). The first substantive action of simplify is to calculate a Gröbner basis F' for the ideal $\langle F\rangle_{\boldsymbol{w}}$ using the variable ordering $\boldsymbol{w}$. If there exists a constant polynomial $c \in F'$ such that $c \neq 0$ then this reveals $\gamma_{\boldsymbol{w}}(F) = \gamma_{\boldsymbol{w}}(F') = \emptyset$ hence $\gamma_{\boldsymbol{w}}(S) = \emptyset$. Otherwise, constrain is invoked if F' contains a polynomial of the form $2^{\omega-j}(w_i + r)$ where $r \in \mathbb{Z}_m[w_{i+1}, \ldots, w_d]$, $0 < j \leq \omega$ and the safety check $\mathsf{safe}(\boldsymbol{W}, w_i, r)$ is satisfied. The added polynomial $w_i - 2^j w + r$ asserts that $w_i + r$ is a multiple of 2^j, which is a direct consequence of $2^{\omega-j}(w_i + r)$. The safety check ensures that the addition of $2^{\omega-j}(w_i + r)$ does not induce a coupling between the variables of $\boldsymbol{w}$, specifically those arising in r, that would compromise the termination argument behind simplify and cover. The safety check is vacuously satisfied if $\mathsf{vars}(r) = \emptyset$.

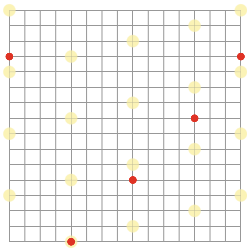
Simplification is used in tandem with splitting, the latter employed by cover only when the former cannot infer new information. When constrain is invoked from cover, two pairs S_0' and S_1' are derived from $S' = \langle \boldsymbol{W}', F'\rangle$ for which $\gamma_{\boldsymbol{w}}(S') = \gamma_{\boldsymbol{w}}(S_0') \cup \gamma_{\boldsymbol{w}}(S_1')$. The pairs S_0' and S_1' are formed by adding $w_i - 2w + 0$ and $w_i - 2w + 1$ to F', which stipulate, respectively, whether w_i takes an even or an odd value. Note, in this case, $\mathsf{constrain}(S', 1, w_i, r)$ is called with $\mathsf{vars}(r) = \emptyset$, hence $\mathsf{safe}(\boldsymbol{W}, w_i, r)$ holds independently of $\boldsymbol{W}$ and w_i and need not be deployed within the body of cover itself. The cover function is then recursively applied to S_0' and S_1' to compute two covers, which are combined by set union. The function returns a singleton set $\{\boldsymbol{W}\}$ when $F = \emptyset$.



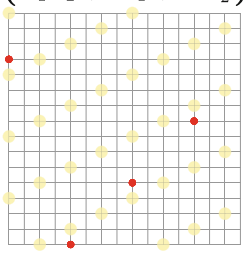
(a) $\langle w_1, w_2 \rangle$
$\left\{ \begin{array}{r} w_1^2 + w_1 + 7w_2^2 + \\ 11w_2 + 12, \\ w_1 w_2 + 4w_1 + 10w_2^2 \end{array} \right\}$



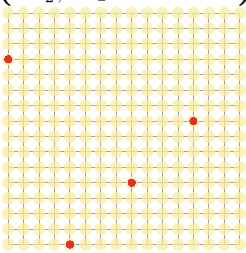
(b) $\langle w_1, 2w_2 \rangle$
$\left\{ \begin{array}{l} w_1^2 + w_1 + 6w_2 + 12, \\ 2w_1 w_2 + 4w_1, \\ 4w_2^2, 8w_2 \end{array} \right\}$



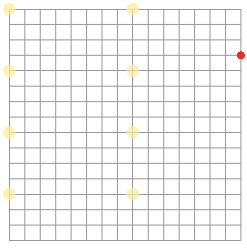
(c) $\langle w_1, 4w_2 \rangle$
$\left\{ \begin{array}{l} w_1^2 + w_1 + 12w_2 + 12, \\ 4w_1 w_2 + 4w_1 \end{array} \right\}$



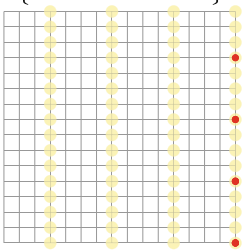
(d) $\langle 2w_1, 4w_2 \rangle$
$\{2w_1 + 12w_2 + 12\}$



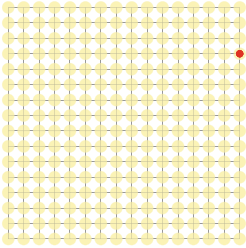
(e) $\langle 4w_2 + 4, 4w_2 \rangle$
$\emptyset$



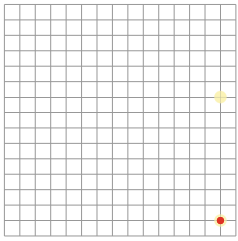
(f) $\langle 2w_1 + 15, 4w_2 \rangle$
$\{2w_1, 4w_2 + 4\}$



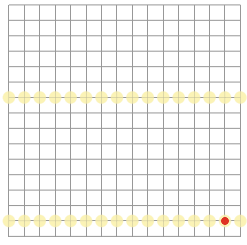
(g) $\langle 15, 4w_2 \rangle$
$\{4w_1 + 4\}$



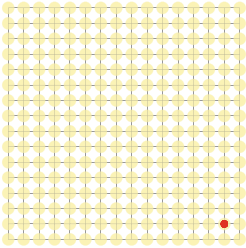
(h) $\langle 15, 12 \rangle$
$\emptyset$



(i) $\langle w_1, 2w_2 + 15 \rangle$
$\{w_1 + 2, 2w_2 + 14\}$



(j) $\langle 14, 2w_2 + 15 \rangle$
$\{2w_2 + 14\}$



(k) $\langle 14, 1 \rangle$
$\emptyset$

Fig. 8. Covering F: $\gamma_{\boldsymbol{y}}(F_n)$ (large, translucent points) and $\gamma_{\boldsymbol{y}}(S_n)$ (small, opaque points) for $S_n = \langle \boldsymbol{W}_n, F_n \rangle$

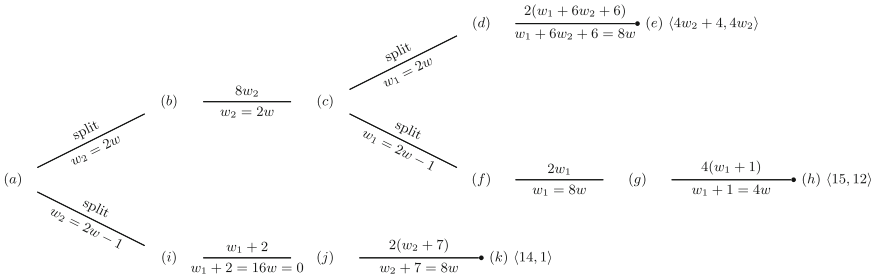


Fig. 9. Covering F: the simplification and splitting actions

Example 19. Figure 9 presents the simplification and splitting actions that arise during a run of the algorithm on the pair $S_a = \langle \boldsymbol{W}_a, F_a \rangle$ introduced in Example 18. The actions are presented as a tree rooted at node a where the leaves, nodes e, h and k, are each decorated with a single polynomial vector. Together these 3 vectors constitute the cover. Figure 8 augments Fig. 9 with details of $S_n = \langle \boldsymbol{W}_n, F_n \rangle$ for each node n of the tree: $\boldsymbol{W}_n$ written above F_n. In each diagram $\gamma_{\boldsymbol{w}}(F_n)$ is represented as large, translucent points and $\gamma_{\boldsymbol{w}}(S_n)$ as small, opaque points. Observe that F_a does not contain any polynomial of the general form $2^{\omega-j}(w_i + r)$ hence cover immediately splits the problem into calculating a cover for $\langle \boldsymbol{W}_b, F_b \rangle$ and a cover for $\langle \boldsymbol{W}_i, F_i \rangle$. Note how splitting doubles a leading constant: $\boldsymbol{W}_a = \boldsymbol{w}$ whereas $\boldsymbol{W}_b = \langle w_1, 2w_2 \rangle$ and $\boldsymbol{W}_i = \langle w_1, 2w_2 + 1 \rangle$. This form of scaling by a power of 2 is a general pattern. By comparing the number of small, opaque points in Fig. 8(a) against those in (b) and (i), observe that the solutions of $\gamma_{\boldsymbol{w}}(S_a)$ are preserved by the split, that is, $\gamma_{\boldsymbol{w}}(S_a) = \gamma_{\boldsymbol{w}}(S_b) \cup \gamma_{\boldsymbol{w}}(S_i)$.

The system F_b contains $8w_2 = 2^{4-1}(w_2 + r)$ where $r = 0$ hence cover deploys simplification to derive $S_c = \langle \boldsymbol{W}_c, F_c \rangle$ from S_b. Since $\mathsf{vars}(r) = \emptyset$ the check $\mathsf{safe}(\boldsymbol{W}, w_i, r)$ is vacuously satisfied. Recall from Example 17 that $\gamma_{\boldsymbol{w}}(S_b) = \gamma_{\boldsymbol{w}}(S_c)$. Observe too how a leading constant is again doubled, with a commensurate doubling in the cardinality of $\gamma_{\boldsymbol{w}}(F_c)$ over $\gamma_{\boldsymbol{w}}(F_b)$. Since F_c does not contain any polynomial $2^{\omega-j}(w_i + r)$ splitting is again applied to give a total of three branches that emanate from a. Observe $F_e = F_g = F_h = \emptyset$ hence the pairs $\langle \boldsymbol{W}_e, F_e \rangle$, $\langle \boldsymbol{W}_g, F_g \rangle$ and $\langle \boldsymbol{W}_h, F_h \rangle$ are degenerate and thereby define the final cover $\{\boldsymbol{W}_e, \boldsymbol{W}_g, \boldsymbol{W}_h\}$ over $\boldsymbol{w}$.

Example 20. Figure 9 illustrates the application of the check $\mathsf{safe}(\boldsymbol{W}, w_i, r)$ within simplify. Observe that $\mathsf{vars}(r) = \emptyset$ in all but one of the simplification steps. For the step that applies $2(w_1 + 6w_2 + 6)$, $r = 6w_2 + 6$ and $\boldsymbol{W} = \langle 2^1 w_1, 2^2 w_2 \rangle$. The polynomial r contains a single term $6w_2$, which contains the single variable w_2. The test $\mathsf{safe}(\boldsymbol{W}, w_1, r)$ thus reduces to a single inequality $k_1 + \mathsf{rank}(6) < k_2$ which is false since $k_1 = 1$, $\mathsf{rank}(6) = 1$ and $k_2 = 2$. Thus safe returns true.

The cover function, and its auxiliaries, are justified by two independent sets of results, the first establishing termination of simplify and cover and the second proving that cover is indeed a cover. The headline results are stated below:

Theorem 2. *simplify and cover terminate.*

Theorem 3. *Let* $F \subseteq \mathbb{Z}_m[\boldsymbol{x}]$ *and* $\mathsf{cover}(F) = \mathcal{W} \subseteq \mathbb{Z}_m^d[\boldsymbol{w}]$. *Then* $\mathcal{W}$ *is a cover of* F *over* $\boldsymbol{w}$.

Example 21. Returning again to F_1 and F_2 of Example 16, cover computes $\mathcal{W}_1 = \{\langle 4w_2+6, 4w_2\rangle, \langle 8, 8w_2+1\rangle, \langle 12, 8w_2+7\rangle\}$ and $\mathcal{W}_2 = \{\langle 8w_1+3, 4w_2+1\rangle\}$ over $\boldsymbol{w} = \langle w_1, w_2\rangle$, where the 3 vectors of $\mathcal{W}_1$ corresponds to A_1, A_2 and A_3 respectively and the single vector constituting $\mathcal{W}_2$ corresponds to A_6 of Fig. 4, but with a different parametric variable w_2 from w_1 used in Example 16.

5.2 Closure

This section explains how a cover provides a vehicle for computing closure. A closed set of polynomials can be represented by different bases, and therefore a relation is introduced to express when one basis represents the closure of another:

Definition 14. *The relation* $\rightarrow_{cl[\boldsymbol{x}]} \subseteq \wp(\mathbb{Z}_m[\boldsymbol{x}])^2$ *is defined* $B \rightarrow_{cl[\boldsymbol{x}]} B'$ *iff* $\uparrow_{\boldsymbol{x}} \langle B\rangle_{\boldsymbol{x}} = \langle B'\rangle_{\boldsymbol{x}}$.

The following lemma provides a method for computing $\uparrow_{\boldsymbol{x}} \langle F\rangle_{\boldsymbol{x}}$ when $\{\boldsymbol{W}\}$ is a singleton cover for F. The lemma is stated by lifting the elimination relation to vectors of variables defined thus $B \rightarrow_{\mathsf{elim}[\epsilon]} B$ and $B \rightarrow_{\mathsf{elim}[y:\boldsymbol{y}]} B''$ iff $B \rightarrow_{\mathsf{elim}[y]} B'$ and $B' \rightarrow_{\mathsf{elim}[\boldsymbol{y}]} B''$. The computational tactic given in the lemma amounts to augmenting null polynomials with d polynomials which equate each variable x_ℓ with W_ℓ and then applying variable elimination:

Lemma 4. *Suppose* $\boldsymbol{W} \in \mathbb{Z}_m[\boldsymbol{w}]^d$, $\top = \langle N\rangle_{\boldsymbol{w}}$ *and* $\{x_1 - W_1, \ldots, x_d - W_d\} \cup N \rightarrow_{elim[\boldsymbol{w}]} B \subseteq \mathbb{Z}_m[\boldsymbol{x}]$. *Then,* $\langle B\rangle_{\boldsymbol{x}} = \alpha_{\boldsymbol{x}}(\{[\![\boldsymbol{W}]\!]_{\boldsymbol{w}}(\boldsymbol{a}) \mid \boldsymbol{a} \in \mathbb{Z}_m^{|\boldsymbol{w}|}\})$.

Example 22. To illustrate this tactic, recall from Example 21 that $\{\boldsymbol{W}\}$ is a cover of F_2 over $\boldsymbol{w} = \langle w_1, w_2\rangle$ where $\boldsymbol{W} = \langle 8w_1+3, 4w_2+1\rangle$. Recall too from Example 4 that $\top = \langle N\rangle_{\boldsymbol{x}}$ where $\boldsymbol{x} = \langle x, y\rangle$. Observe $N' = N[x \mapsto w_1, y \mapsto w_2]$ is also a set of nulls, albeit over $\mathbb{Z}_m[\boldsymbol{w}]$. Let $p'_i = p_i[x \mapsto w_1, y \mapsto w_2]$ using the abbreviations of Example 4. Then $\mathsf{gb}_{\boldsymbol{w}:\boldsymbol{x}}(\{x - W_1, y - W_2\} \cup N']) = B$ where

$$B = \left\{ \begin{array}{c} p'_1, \quad p'_2, \quad p'_3, \quad p'_4, \quad 2w_1y + 6w_1 + w_2x + w_2 + x + 3y + 10, \\ w_1^4y + w_1^4 + 4w_1^3 + w_1^2y + 5w_1^2 + 4w_1 + w_2x + w_2 + 3y + 13, \\ w_1^2w_2y + 3w_1^2w_2 + w_1w_2y + 3w_1w_2, \quad w_1x + 5w_1, \quad 8w_1 + x + 13, \\ \hline w_2^3y + 3w_2^3 + w_2^2y + 3w_2^2, w_2^2x + w_2^2 + w_2x + w_2y + y + 15, \\ p'_5, \quad p'_6, \quad 2w_2y + 2w_2 + y + 15, \quad 4w_2 + 3y + 13, \\ \hline x^2 + 7, \quad xy + x + y + 9, \quad 2x + 10, \quad y^2 + 2y + 13, \quad 4y + 12 \end{array} \right\}$$

The three regions delineate polynomials depending on both w_1 and w_2 (top), w_2 but not w_1 (middle) and neither w_1 nor w_2 (bottom). It thus follows that $\{x - W_1, y - W_2\} \cup N' \rightarrow_{\mathsf{elim}[w_1]} B'$ where

$$B' = \left\{ \begin{array}{c} w_2^3y + 3w_2^3 + w_2^2y + 3w_2^2, w_2^2x + w_2^2 + w_2x + w_2y + y + 15, \\ p'_5, \quad p'_6, \quad 2w_2y + 2w_2 + y + 15, \quad 4w_2 + 3y + 13, \\ \hline x^2 + 7, \quad xy + x + y + 9, \quad 2x + 10, \quad y^2 + 2y + 13, \quad 4y + 12 \end{array} \right\}$$

B' is a Gröbner basis (with respect to $\prec_{\langle w,x,y\rangle}$), hence $B' \rightarrow_{\mathsf{elim}[w_2]} B''$ where

$$B'' = \{x^2 + 7, \quad xy + x + y + 9, \quad 2x + 10, \quad y^2 + 2y + 13, \quad 4y + 12\}$$

Composing the two eliminations yields $\{x - W_1, y - W_2\} \cup N' \rightarrow_{\mathsf{elim}[\boldsymbol{w}]} B''$. Note that it is only necessary to compute a single Gröbner basis to derive B''. Observe that each polynomial of B'' satisfies the points of $\gamma_{\boldsymbol{x}}(F_2)$ illustrated in Fig. 4(g).

The following theorem generalises this tactic to arbitrary covers:

Theorem 4. *Let $B \subseteq \mathbb{Z}_m[\boldsymbol{x}]$, $\top = \langle N\rangle_{\boldsymbol{w}}$ and $\mathcal{W} \subseteq \mathbb{Z}_m[\boldsymbol{w}]^d$ be a cover for B over $\boldsymbol{w}$. Suppose for each $\boldsymbol{W} \in \mathcal{W}$, $\{x_1 - W_1, \ldots, x_d - W_d\} \cup N \rightarrow_{\mathit{elim}[\boldsymbol{w}]} B_{\boldsymbol{W}}$ and $\langle B'\rangle_{\boldsymbol{x}} = \bigsqcup_{\boldsymbol{W}\in\mathcal{W}} \langle B_{\boldsymbol{W}}\rangle_{\boldsymbol{x}}$. Then, $B \rightarrow_{\mathit{cl}[\boldsymbol{x}]} B'$.*

Example 23. Now recall from Example 19 that $\{\boldsymbol{W}_e, \boldsymbol{W}_h, \boldsymbol{W}_k\}$ is a cover of

$$F = \left\{x^2 + x + 7y^2 + 11y + 12, \quad xy + 4x + 10y^2\right\}$$

over $\boldsymbol{w} = \langle w_1, w_2\rangle$ where $\boldsymbol{W}_e = \langle 4w_2 + 4, 4w_2\rangle$, $\boldsymbol{W}_h = \langle 15, 12\rangle$ and $\boldsymbol{W}_k = \langle 14, 1\rangle$. To apply the theorem, $B_{\boldsymbol{W}_e}$ is derived by $\{x - (4w_2 + 4), y - 4w_2\} \cup N \rightarrow_{\mathsf{elim}[\boldsymbol{w}]} B_{\boldsymbol{W}_e}$. Since $\boldsymbol{W}_e$ depends only on w_2, $B_{\boldsymbol{W}_e}$ can be computed by $\{x - (4w_2 + 4), y - 4w_2\} \cup N' \rightarrow_{\mathsf{elim}[w_2]} B_{\boldsymbol{W}_e}$ where $\top = \langle N'\rangle_{\langle w_2\rangle}$. To that end, note $\mathsf{gb}_{\langle w_2,x,y\rangle}(\{x - (4w_2 - 4), y - 4w_2\} \cup N') = B'_{\boldsymbol{W}_e}$ where

$$B'_{\boldsymbol{W}_e} = \left\{ \begin{array}{c} w_2^6 + w_2^5 + w_2^4 + 3w_2^3 + w_2^2 y + 2w_2^2 + w_2 y, \\ 2w_2^4 + w_2^2 y + 2w_2^2 + w_2 y + y, \quad w_2^3 y + w_2 y + 2y, \\ 2w_2 y + 2y, \quad 4w_2 + 3y, \quad x + 3y + 12, \quad y^2, \quad 4y \end{array} \right\}$$

thus $B_{\boldsymbol{W}_e} = \{x + 3y + 12, y^2, 4y\}$ is computed avoiding nulls containing w_1.

The bases $B_{\boldsymbol{W}_h}$ and $B_{\boldsymbol{W}_k}$ can be derived without recourse to elimination or any nulls since $\boldsymbol{W}_h$ and $\boldsymbol{W}_k$ are independent of w_1 and w_2 hence put

$$B_{\boldsymbol{W}_h} = \{x{-}15, y{-}12\} = \{x{+}1, y{+}4\} \qquad B_{\boldsymbol{W}_k} = \{x{-}14, y{-}1\} = \{x{+}2, y{+}15\}$$

By Theorem 4 $\uparrow_{\boldsymbol{x}} \langle F\rangle_{\boldsymbol{x}} = \langle B\rangle_{\boldsymbol{x}}$ where $\langle B\rangle_{\boldsymbol{x}} = \langle B_{\boldsymbol{W}_e}\rangle_{\boldsymbol{x}} \sqcup \langle B_{\boldsymbol{W}_h}\rangle_{\boldsymbol{x}} \sqcup \langle B_{\boldsymbol{W}_k}\rangle_{\boldsymbol{x}}$ giving

$$B = \left\{x^2 + x + 7y^2 + 11y + 12, \quad xy + 4x + 10y^2, \quad y^3 + 7y^2 + 8y, \quad 4y^2 + 12y\right\}$$

All the polynomials of B satisfy the points $\gamma_{\boldsymbol{x}}(F)$ plotted in Fig. 7(a). Observe

$$F' = \left\{x^2 + x + 7y^2 + 11y + 12, \quad xy + 4x + 10y^2, \quad y^3 + 7y^2 + 8y\right\}$$

is a Gröbner basis for $\langle F\rangle_{\boldsymbol{x}}$ with respect to $\prec_{\boldsymbol{x}}$. Since $4y^2 + 12y$ is irreducible by F' it follows $4y^2 + 12y \notin \langle F\rangle_{\boldsymbol{x}}$ which is why closure augments F with $4y^2 + 12y$.

5.3 Meet

Despite the central importance of meet, this section is relatively short, since the following proposition demonstrates how meet can be reduced to closure:

Proposition 8. *Let* $\langle B_1\rangle_{\boldsymbol{x}}, \langle B_2\rangle_{\boldsymbol{x}} \in \mathit{MPAD}_m[\boldsymbol{x}]$. *If* $B_1 \cup B_2 \rightarrow_{\mathit{cl}[\boldsymbol{x}]} B$ *then* $\langle B_1\rangle_{\boldsymbol{x}} \sqcap \langle B_2\rangle_{\boldsymbol{x}} = \langle B\rangle_{\boldsymbol{x}}$.

Example 24. Consider $F_3, F_4 \subseteq \mathbb{Z}_{16}[x, y]$ where $F_3 = \{x^2+x+7y^2+11y+12\}$ and $F_4 = \{xy+4x+10y^2\}$ and let $F = F_3 \cup F_4$. The solution sets $\gamma_{\boldsymbol{x}}(F)$, $\gamma_{\boldsymbol{x}}(F_3)$ and $\gamma_{\boldsymbol{x}}(F_4)$ are plotted in Figs. 7(a), (b) and (c) respectively. The diamond points in Figs. 7(b) and (c) are those contained in both $\gamma_{\boldsymbol{x}}(F_3)$ and $\gamma_{\boldsymbol{x}}(F_4)$ and show $\gamma_{\boldsymbol{x}}(F) = \gamma_{\boldsymbol{x}}(F_1) \cap \gamma_{\boldsymbol{x}}(F_2)$. Now, Example 23 shows $F \rightarrow_{\mathsf{cl}[\boldsymbol{x}]} B$ where

$$B = \left\{x^2 + x + 7y^2 + 11y + 12, \quad xy + 4x + 10y^2, \quad y^3 + 7y^2 + 8y, \quad 4y^2 + 12y\right\}$$

thus $\langle F_3\rangle_{\boldsymbol{x}} \sqcap \langle F_4\rangle_{\boldsymbol{x}} = \langle B\rangle_{\boldsymbol{x}}$. As noted in Example 23, $4y^2 + 12y \notin \langle F\rangle_{\boldsymbol{x}}$ hence $\langle F\rangle_{\boldsymbol{x}} \neq \langle B\rangle_{\boldsymbol{x}}$.

6 Forwards Analysis of Polynomial Programs

In this section, the class of polynomial programs is introduced for which a concrete semantics is defined over sets of points drawn from $\mathbb{Z}_m^d$. The corresponding abstract semantics over MPAD defines a forwards analysis. The development builds to show the soundness of the analysis, as well as state a precision result for programs consisting solely of polynomial assignments, non-deterministic assignments and non-deterministic branching. The section concludes with an illustrative example for a program which computes the modular inverse.

6.1 Polynomial Programs

Let $\boldsymbol{x} = \langle x_1, \ldots, x_d\rangle$ denote a vector of program variables. A polynomial program over $\boldsymbol{x}$ is a graph $G = \langle N, E, n^*\rangle$ where N is a finite set of program points, $E \subseteq N \times \mathsf{Stmt} \times N$ is a finite set of annotated edges and $n^* \in N$ is the entry point into G. The set Stmt of program statements is defined:

$$x_j \coloneqq p \quad | \quad x_j \coloneqq * \quad | \quad \mathsf{assume}\ (p = 0) \quad | \quad \mathsf{assume}\ (p \neq 0)$$

where $x_j \coloneqq *$ and $x_j \coloneqq p$ denote, respectively, non-deterministic assignment to the variable x_j and polynomial assignment to x_j for some $p \in \mathbb{Z}_m[\boldsymbol{x}]$. The assume statements for $p = 0$ and $p \neq 0$ provide a linguistic abstraction for positive and negative guards, respectively expressing that p is satisfied, and conversely p is not satisfied, by an assignment to $\boldsymbol{x}$.

6.2 Concrete Semantics

To define the concrete semantics, let $\boldsymbol{a}[j \mapsto c] = \langle a_1, \ldots, a_{j-1}, c, a_{j+1}, \ldots, a_d\rangle$ for $\boldsymbol{a} \in \mathbb{Z}_m^d$, $c \in \mathbb{Z}_m$ and j a variable index denote a vector update. The concrete

semantics is then formulated in terms of a set of (concrete) transfer functions $[\![s]\!] : \wp(\mathbb{Z}_m^d) \to \wp(\mathbb{Z}_m^d)$, one for each program statement s, defined as follows:

$$\begin{aligned}
[\![x_j \coloneqq p]\!](A) &= \{(\boldsymbol{a})[j \mapsto c] \mid \boldsymbol{a} \in A, c = [\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a})\} \\
[\![x_j \coloneqq *]\!](A) &= \{(\boldsymbol{a})[j \mapsto c] \mid \boldsymbol{a} \in A, c \in \mathbb{Z}_m\} \\
[\![\mathsf{assume}\ (p = 0)]\!](A) &= \{\boldsymbol{a} \in A \mid [\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a}) = 0\} \\
[\![\mathsf{assume}\ (p \neq 0)]\!](A) &= \{\boldsymbol{a} \in A \mid [\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a}) \neq 0\}
\end{aligned}$$

Observe that the function space $N \to \wp(\mathbb{Z}_m^d)$ is ordered point-wise: $\theta \sqsubseteq \theta'$ iff $\theta(n) \subseteq \theta'(n)$ for all $n \in N$. Thus the concrete semantics can be defined as follows:

Definition 15. *The concrete semantics for $G = \langle N, E, n^* \rangle$ is the least map $\theta : N \to \wp(\mathbb{Z}_m^d)$ satisfying:*

- $\mathbb{Z}_m^d \subseteq \theta(n^*)$
- $[\![s]\!](\theta(n)) \subseteq \theta(n')$ *for all* $\langle n, s, n' \rangle \in E$

6.3 Abstract Semantics

Analogous to the concrete semantics, the abstract semantics for $G = \langle N, E, n^* \rangle$ is defined in terms of a set of (abstract) transfer functions. For a statement s, $[\![s]\!] : \mathsf{MPAD}_m[\boldsymbol{x}] \to \mathsf{MPAD}_m[\boldsymbol{x}]$ is defined thus:

$$\begin{aligned}
[\![x_j \coloneqq p]\!](P) &= \{q \in \mathbb{Z}_m[\boldsymbol{x}] \mid q[p/x_j] \in P\} \\
[\![x_j \coloneqq *]\!](P) &= \uparrow_{\boldsymbol{x}} \mathsf{elim}[x_j](P) \\
[\![\mathsf{assume}\ (p = 0)]\!](P) &= \uparrow_{\boldsymbol{x}} (\{p\} \cup P) \\
[\![\mathsf{assume}\ (p \neq 0)]\!](P) &= \bigsqcup_{k=1}^{\omega} [\![\mathsf{assume}\ (2^{\omega-k}p + 2^{\omega-1} = 0)]\!](P)
\end{aligned}$$

Here, the notation $q[p/x_j]$ denotes the polynomial constructed by substituting p for every instance of x_j in q. To comprehend the encoding for $\mathsf{assume}(p \neq 0)$, suppose $\boldsymbol{a} \in \mathbb{Z}_m$ such that $[\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a}) \neq 0$. Observe there exists some $1 \leq k \leq \omega$ such that $[\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a})$ has 1 in its k-th lowest bit position and 0 in all lower bits. Therefore $[\![2^{\omega-k}p + 2^{\omega-1}]\!]_{\boldsymbol{x}}(\boldsymbol{a}) = 0$. Conversely, if $[\![2^{\omega-k}p + 2^{\omega-1}]\!]_{\boldsymbol{x}}(\boldsymbol{a}) = 0$ then $[\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a})$ has 1 in its k-th lowest bit position hence $[\![p]\!]_{\boldsymbol{x}}(\boldsymbol{a}) \neq 0$.

The function space $N \to \mathsf{MPAD}_m[\boldsymbol{x}]$ is likewise ordered point-wise: $\sigma \sqsubseteq \sigma'$ iff $\sigma(n) \sqsubseteq \sigma'(n)$ for all $n \in N$, allowing the abstract semantics to be defined thus:

Definition 16. *The abstract semantics for $G = \langle N, E, n^* \rangle$ is the least map $\sigma : N \to \mathit{MPAD}_m[\boldsymbol{x}]$ satisfying:*

- $\top \sqsubseteq \sigma(n^*)$
- $[\![s]\!](\theta(n)) \sqsubseteq \sigma(n')$ *for all* $\langle n, s, n' \rangle \in E$

Since MPAD is finite, the abstract semantics can be concretely computed by fixed-point iteration; an example of such a procedure is illustrated in Sect. 6.5. The relationship between the concrete and abstract semantics is developed in the following section. The following result details how the abstract transfer functions are actually computed:

Proposition 9. *Let* $\langle B\rangle_{\boldsymbol{x}} \in \mathit{MPAD}_m[\boldsymbol{x}]$ *and* $w \notin \mathsf{vars}(B)$. *Then:*

- *If* $B \cup \{w - p\} \rightarrow_{\mathit{elim}[x_j]} B'$ *and* $B' \cup \{x_j - w\} \rightarrow_{\mathit{elim}[w]} B''$ *then it follows* $[\![x_j \coloneqq p]\!](\langle B\rangle_{\boldsymbol{x}}) = \langle B''\rangle_{\boldsymbol{x}}$.
- *If* $B \rightarrow_{\mathit{elim}[x_j]} B'$ *and* $B' \rightarrow_{\mathit{cl}[\boldsymbol{x}]} B''$ *then* $[\![x_j \coloneqq *]\!](\langle B\rangle_{\boldsymbol{x}}) = \langle B''\rangle_{\boldsymbol{x}}$.
- *If* $\{p\} \cup B \rightarrow_{\mathit{cl}[\boldsymbol{x}]} B'$ *then* $[\![\mathsf{assume}\ (p = 0)]\!](\langle B\rangle_{\boldsymbol{x}}) = \langle B'\rangle_{\boldsymbol{x}}$.

6.4 Correctness and Precision

Key to establishing correctness and precision of the analysis are the following results. The first elucidates the relationship between concrete and abstract join using the abstraction map $\alpha_{\boldsymbol{x}} : \wp(\mathbb{Z}_m^d) \rightarrow \mathsf{MPAD}_m[\boldsymbol{x}]$ introduced in Proposition 3:

Proposition 10. *Suppose* $P_1, P_2 \in \mathit{MPAD}_m[\boldsymbol{x}]$ *where* $P_1 = \alpha_{\boldsymbol{x}}(A_1), P_2 = \alpha_{\boldsymbol{x}}(A_2)$ *for some* $A_1, A_2 \subseteq \mathbb{Z}_m^d$. *Then* $P_1 \sqcup P_2 = \alpha_{\boldsymbol{x}}(A_1 \cup A_2)$.

The second result demonstrates soundness of each of the abstract transfer functions, as well as optimality for the two assignment operations:

Proposition 11. *Let* $A \subseteq \mathbb{Z}_m^d$ *and* $P = \alpha_{\boldsymbol{x}}(A)$ *so that* $P \in \mathit{MPAD}_m[\boldsymbol{x}]$. *Then*

$$\begin{aligned}
\alpha_{\boldsymbol{x}}([\![x_j \coloneqq p]\!](A)) &= [\![x_j \coloneqq p]\!](P)\\
\alpha_{\boldsymbol{x}}([\![x_j \coloneqq *]\!](A)) &= [\![x_j \coloneqq *]\!](P)\\
\alpha_{\boldsymbol{x}}([\![\mathsf{assume}\ (p = 0)]\!](A)) &\sqsubseteq [\![\mathsf{assume}\ (p = 0)]\!](P)\\
\alpha_{\boldsymbol{x}}([\![\mathsf{assume}\ (p \neq 0)]\!](A)) &\sqsubseteq [\![\mathsf{assume}\ (p \neq 0)]\!](P)
\end{aligned}$$

With these results in the place, the following theorem can be demonstrated:

Theorem 5. *If the concrete and the abstract semantics for* $G = \langle N, E, n^*\rangle$ *are* $\theta : N \rightarrow \wp(\mathbb{Z}_m^d)$ *and* $\sigma : N \rightarrow \mathit{MPAD}_m[\boldsymbol{x}]$ *respectively then:*

$$\alpha_{\boldsymbol{x}}(\theta(n)) \sqsubseteq \sigma(n) \text{ for all } n \in N$$

If G *is free from* $\mathsf{assume}\ (p = 0)$ *and* $\mathsf{assume}\ (p \neq 0)$ *statements then:*

$$\alpha_{\boldsymbol{x}}(\theta(n)) = \sigma(n) \text{ for all } n \in N$$

In particular, MPAD provides a sound analysis for polynomial programs, and moreover finds all modular polynomial invariants for programs consisting of polynomial and non-deterministic assignments and non-deterministic branching.

6.5 Illustrative Example

To illustrate how MPAD can be applied, consider the algorithm [34] listed in Fig. 10(a). The algorithm computes the multiplicative inverse of an (odd) modular integer $a \in \mathbb{Z}_m$. The variables x, y and a all store a ω-bit (unsigned) integer. The algorithm is abstracted by the polynomial program represented in Fig. 10(b) where $\boldsymbol{x} = \langle x, y, a\rangle$, the nodes are $N = \{0, \ldots, 7\}$ and the entry node is 0. Each

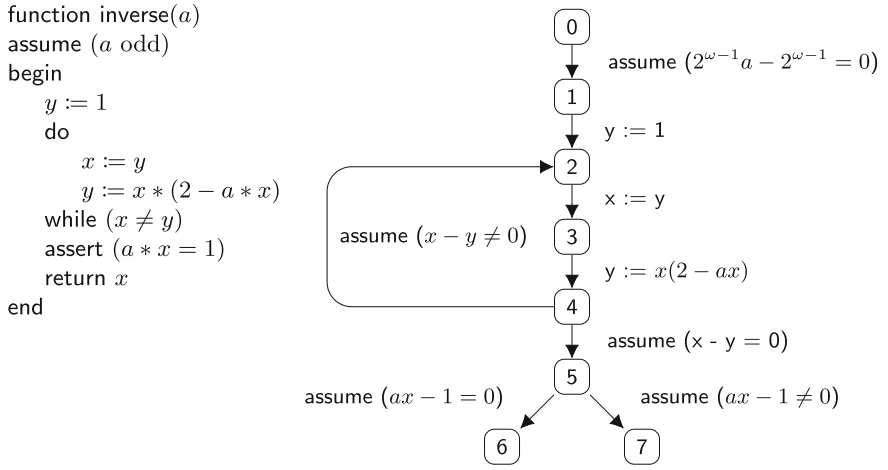


Fig. 10. An algorithm and flow graph for computing the multiplicative inverse

edge is decorated with a polynomial assignment or an assumption involving a polynomial equality or a polynomial disequality.

The statement assume $(a$ odd$)$ is rendered as assume $(2^{\omega-1}a - 2^{\omega-1} = 0)$, where the (linear) polynomial $2^{\omega-1}a - 2^{\omega-1} = 0$ expresses that a is odd. The control-flow for the **do** ... **while** is represented as two edges decorated with assume $(x - y \neq 0)$ and assume $(x - y = 0)$, which, respectively, encode the loop condition $x \neq y$ and its negation. The control flow for the assert statement is expressed through two edges decorated with assume $(ax - 1 = 0)$ and assume $(ax - 1 \neq 0)$, where the node 7 is reached if the assertion fails.

Figure 11 presents each σ_k computed by a work-list algorithm primed with the edges that flow from 0. The second column displays the worklist w_k. The selected edge $\langle n, n' \rangle \in w_k$ is always the first listed in w_k. For instance, at step 4, the edge $\langle 4, 2 \rangle$ is selected, rather than $\langle 4, 5 \rangle$. The third column displays σ_{k+1} as a function of the σ_k: σ_k if no update occurred, else $\sigma_k[n' \mapsto P]$ where $P \in \mathsf{MPAD}_m[\boldsymbol{x}]$. Polynomials that appear multiply are referenced with a label: p_a, p_b, etc. Most significantly, the table demonstrates $\sigma_{14}(7) = \bot$ thus assert $(a * x = 1)$ must succeed (this can be seen from $ax - 1 \rightarrow^*_{\prec, \sigma_{13}(5)} 0$). No other abstract domain can verify this code because the invariants are both polynomial and modular and the analysis requires polynomial guards; indeed the manual proof of correctness of the algorithm [34] relies on both polynomial manipulation and observing $a^2(2^\omega) = 0 \mod 2^\omega$.

Since a positive polynomial guard is modelled by meet, detailed commentaries are only given for a polynomial assignment and a negative guard:

Polynomial Assignment. When $k = 1$, the edge $\langle 1, 2 \rangle$ is selected, corresponding to the statement $y := 1$. From the table it follows $\sigma_1(1) = P_0 = \langle B_0 \rangle_{\boldsymbol{x}}$. To model the assignment, first the basis B_0 is adjoined with the polynomial $w - 1$. Here,

k	w_k	σ_{k+1}
0	$\{\langle 0,1\rangle\}$	$\sigma_0[1 \mapsto \langle x^4a + x^4 + 2x^3a + 2x^3 + 3x^2a + 3x^2 + 2xa + 2x\ (p_a),$ $2x^2y^2a + 2x^2y^2 + 2x^2ya +$ $2x^2y + 2xy^2a + 2xy^2 + 2xya + 2xy,$ $x^2a^2 + 7x^2 + xa^2 + 7x\ (p_b), 4x^2a + 4x^2 + 4xa + 4x\ (p_c),$ $y^4a + y^4 + 2y^3a + 2y^3 + 3y^2a + 3y^2 + 2ya + 2y,$ $y^2a^2 + 7y^2 + ya^2 + 7y, 4y^2a + 4y^2 + 4ya + 4y,$ $a^3 + a^2 + 7a + 7\ (p_d), 2a^2 + 14\ (p_e), 8a + 8\ (p_f)\rangle_{\boldsymbol{x}}]$
1	$\{\langle 1,2\rangle\}$	$\sigma_1[2 \mapsto \langle p_a, p_b, p_c, y + 15\ (p_g), p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
2	$\{\langle 2,3\rangle\}$	$\sigma_2[3 \mapsto \langle x + 15\ (p_h), p_g, p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
3	$\{\langle 3,4\rangle\}$	$\sigma_3[4 \mapsto \langle p_h, y + a + 14, p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
4	$\{\langle 4,2\rangle, \langle 4,5\rangle\}$	$\sigma_4[2 \mapsto \langle p_a, p_b, p_c, xy + 15x + 7y + 9, y^2 + ya + 5y + 7a + 2\ (p_i),$ $ya^2 + 7y + a^2 + 7\ (p_j), 2ya + 2y + 6a + 6\ (p_k),$ $8y + 8\ (p_l), p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
5	$\{\langle 2,3\rangle, \langle 4,5\rangle\}$	$\sigma_5[3 \mapsto \langle x + 7y + 8\ (p_m), p_i, p_j, p_k, p_l, p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
6	$\{\langle 3,4\rangle, \langle 4,5\rangle\}$	$\sigma_6[4 \mapsto \langle x^2 + 2x + 3y + 3a + 7\ (p_n),$ $xy + 3x + a + 11, xa + 3x + y + 11,$ $4x + 2y + 2a + 8\ (p_o), y^2 + 2y + a^2 + 6a + 6\ (p_q),$ $ya + y + a^2 + 7a + 6\ (p_r), 4y + 4a + 8\ (p_s), p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
7	$\{\langle 4,2\rangle, \langle 4,5\rangle\}$	$\sigma_7[2 \mapsto \langle p_a, x^2y + 7x^2 + 8x + 7y + 9, p_b, p_c, xy^2 + 15x + y^2 + 15,$ $xya + xy + 7xa + 7x + ya + y + 7a + 7,$ $2xy + 14x + y^2 + ya + 3y + 7a + 4,$ $y^3 + y^2 + 7y + 7\ (p_t), y^2a + y^2 + 7a + 7\ (p_u),$ $2y^2 + 14\ (p_v), p_j, p_k, p_l, p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
8	$\{\langle 2,3\rangle, \langle 4,5\rangle\}$	$\sigma_8[3 \mapsto \langle p_m, p_t, p_u, p_v, p_j, p_k, p_l, p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
9	$\{\langle 3,4\rangle, \langle 4,5\rangle\}$	$\sigma_9[4 \mapsto \langle p_n,$ $xy + xa + 2x + 3y + 3a + 6, xa^2 + 3x + 2y + a^2 + 2a + 7,$ $2xa + 2x + 6a + 6, p_o, p_q, p_r, p_s, p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
10	$\{\langle 4,2\rangle, \langle 4,5\rangle\}$	σ_{10}
11	$\{\langle 4,5\rangle\}$	$\sigma_{11}[5 \mapsto \langle x + a^2 + 7a + 7\ (p_w), y + a^2 + 7a + 7\ (p_x), p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
12	$\{\langle 5,6\rangle, \langle 5,7\rangle\}$	$\sigma_{12}[6 \mapsto \langle p_w, p_x, p_d, p_e, p_f\rangle_{\boldsymbol{x}}]$
13	$\{\langle 5,7\rangle\}$	σ_{13}

Fig. 11. Updates to the state map

w is a new variable that represents the value of y after the assignment and the polynomial $w - 1$ expresses that this value must equal 1. Then, y is eliminated from $B_0 \cup \{w - 1\}$, to reflect that y is overwritten during the assignment. This elimination step is achieved in two phases. First, a Gröbner basis is computed for $\langle B_0 \cup \{w - 1\}\rangle_{\boldsymbol{x}}$ with respect to a lexicographical ordering $\langle y, x, w, a\rangle$ over the variables, yielding

$$\left\{\begin{array}{c} y^4a + y^4 + 2y^3a + 2y^3 + 3y^2a + 3y^2 + 2ya + 2y, \\ 2y^2x^2a + 2y^2x^2 + 2y^2xa + 2y^2x + 2yx^2a + 2yx^2 + 2yxa + 2yx, \\ y^2a^2 + 7y^2 + ya^2 + 7y, \quad 4y^2a + 4y^2 + 4ya + 4y, \\ x^4a + x^4 + 2x^3a + 2x^3 + 3x^2a + 3x^2 + 2xa + 2x, \quad x^2a^2 + 7x^2 + xa^2 + 7x, \\ 4x^2a + 4x^2 + 4xa + 4x, \quad w + 15, \quad a^3 + a^2 + 7a + 7, \quad 2a^2 + 14, \quad 8a + 8 \end{array}\right\}$$

Then, all polynomials involving y are deleted:

$$B_0' = \left\{ \begin{matrix} x^4a + x^4 + 2x^3a + 2x^3 + 3x^2a + 3x^2 + 2xa + 2x, \\ x^2a^2 + 7x^2 + xa^2 + 7x, \quad 4x^2a + 4x^2 + 4xa + 4x, \\ w + 15, \quad a^3 + a^2 + 7a + 7, \quad 2a^2 + 14, \quad 8a + 8 \end{matrix} \right\}$$

the Gröbner basis ensuring that this deletion does not lose information. To finalise the assignment, a Gröbner basis is computed for $\langle B_0' \cup \{w - y\}\rangle_{\boldsymbol{x}}$ with respect to the lexicographical ordering $\langle w, y, x, a\rangle$, then all constraints containing w are deleted, yielding:

$$B_1 = \left\{ \begin{matrix} x^4a + x^4 + 2x^3a + 2x^3 + 3x^2a + 3x^2 + 2xa + 2x, \\ x^2a^2 + 7x^2 + xa^2 + 7x, \quad 4x^2a + 4x^2 + 4xa + 4x, \\ y + 15, \quad a^3 + a^2 + 7a + 7, \quad 2a^2 + 14, \quad 8a + 8 \end{matrix} \right\}$$

Negative Polynomial Guard. When $k = 4$, the edge $\langle 4, 2\rangle$ is selected, the edge corresponding to the statement $\mathsf{assume}\ (x - y \neq 0)$. From the table it follows $\sigma_3(4) = \langle B_3\rangle_{\boldsymbol{x}}$ where

$$B_3 = \left\{x + 15, \quad y + a + 14, \quad a^3 + a^2 + 7a + 7, \quad 2a^2 + 14, \quad 8a + 8\right\}$$

To effect the operation, closure is separately applied to four bases:

$$\begin{array}{lll} B_3 \cup \{8(x-y)+8\} & \rightarrow_{\mathsf{cl}[\boldsymbol{x}]} & B_{4,1} = \{1\} \\ B_3 \cup \{4(x-y)+8\} & \rightarrow_{\mathsf{cl}[\boldsymbol{x}]} & B_{4,2} = \{x + 15,\ y + a + 14,\ a^2 + 2a + 1,\ 4a + 4\} \\ B_3 \cup \{2(x-y)+8\} & \rightarrow_{\mathsf{cl}[\boldsymbol{x}]} & B_{4,3} = \{x + 15,\ y + a + 14,\ a^2 + 7,\ 2a + 6\} \\ B_3 \cup \{(x-y)+8\} & \rightarrow_{\mathsf{cl}[\boldsymbol{x}]} & B_{4,4} = \{x + 15,\ y + 7,\ a + 7\} \end{array}$$

The intuition is that each $\gamma_{\boldsymbol{x}}(B_{4,k})$ is the subset of $\boldsymbol{a} \in \gamma_{\boldsymbol{x}}(B_3)$ for which the k least-significant bits of $[\![x - y]\!]_{\boldsymbol{x}}(\boldsymbol{a})$ store the value 2^{k-1}. Thus $\gamma_{\boldsymbol{x}}(B_{4,1})$ is the subset of $\boldsymbol{a} \in \gamma_{\boldsymbol{x}}(B_3)$ for which the least bit of $[\![x - y]\!]_{\boldsymbol{x}}(\boldsymbol{a})$ is $2^0 = 1$; $\gamma_{\boldsymbol{x}}(B_{4,2})$ is the subset for which the 2 least bits of $[\![x - y]\!]_{\boldsymbol{x}}(\boldsymbol{a})$ store $2^1 = 2$, etc. Since $[\![x - y]\!]_{\boldsymbol{x}}(\boldsymbol{a}) \neq 0$ holds precisely when at least one bit is set, it follows $P_4 = \bigsqcup_{k=1}^{4} \langle B_{4,k}\rangle_{\boldsymbol{x}} \in \mathsf{MPAD}_m[\boldsymbol{x}]$ satisfies the property above. In fact, in this case $P_4 = P_3$, hence the abstract execution of $\mathsf{assume}\ (x - y \neq 0)$ does not strengthen the polynomial constraints even though $B_{4,1} = \{1\}$ reveals that the difference between x and y is never odd.

6.6 Implementation

Buchberger's algorithm, like many in symbolic computation, has poor worst-case complexity [19]. Performance can be dramatically improved, however, by factoring out redundant s-polynomial calculations (and the ensuing reductions) using the so-called Gebauer and Möller rules [13]. To reestablish these rules for modular polynomials, let $B = \{p_1, \ldots, p_s\} \subseteq \mathbb{Z}_m[\boldsymbol{x}] \setminus \{0\}$ and put $\boldsymbol{p} = \langle p_1, \ldots, p_s\rangle$ and $\boldsymbol{t} = \langle \mathsf{lt}_{\prec}(p_1), \ldots, \mathsf{lt}_{\prec}(p_s)\rangle$. A vector $\boldsymbol{q} \in \mathbb{Z}_m[\boldsymbol{x}]^s$ is a syzygy of $\boldsymbol{t}$ iff $\boldsymbol{q} \cdot \boldsymbol{t} = 0$. The set $\mathsf{syz}(\boldsymbol{t})$ of syzygies of $\boldsymbol{t}$ forms a module. It can be shown

[13] that if $\boldsymbol{q} \cdot \boldsymbol{p} \rightarrow_{\prec,B} 0$ for all syzygies $\boldsymbol{q}$ in a module-basis for $\mathsf{syz}(\boldsymbol{t})$, then B is a Gröbner basis. Letting $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_s\}$ denote the standard basis for $\mathbb{Z}_m[\boldsymbol{x}]^s$, $k_i = \mathsf{rank}(\mathsf{lc}_\prec(p_i))$ and $t_{i,j} = \mathsf{lcm}(t_i, t_j)/t_i$, the principle syzygies π_i and $\pi_{i,j}$ are defined:

$$\pi_i = 2^{\omega - k_i}\boldsymbol{e}_i \quad \text{and} \quad \pi_{i,j} = t_{i,j}\boldsymbol{e}_i - t_{j,i}\boldsymbol{e}_j \text{ where } 1 \leq i < j \leq s$$

The following result yields a condition for detecting redundant principle syzygies:

Proposition 12. *Given $\boldsymbol{t} = \langle t_1, \ldots, t_s \rangle$ be a vector of non-zero terms. Then*

$$\frac{\mathit{lcm}(t_i, t_j, t_k)}{\mathit{lcm}(t_i, t_j)}\boldsymbol{\pi}_{i,j} + \frac{\mathit{lcm}(t_i, t_j, t_k)}{\mathit{lcm}(t_j, t_k)}\boldsymbol{\pi}_{j,k} + \frac{\mathit{lcm}(t_i, t_j, t_k)}{\mathit{lcm}(t_k, t_i)}\boldsymbol{\pi}_{k,i} = 0$$

and, in particular, if t_k divides $\mathsf{lcm}(t_i, t_j)$ then $\boldsymbol{\pi}_{i,j}$ is in the submodule generated by $\boldsymbol{\pi}_{j,k}$ and $\boldsymbol{\pi}_{k,i}$

Principle syzygies align with S-polynomials: π_i and $\pi_{i,j}$ correspond to $\mathsf{S}_\prec(p_i, p_i)$ and $\mathsf{S}_\prec(p_i, p_j)$ respectively. Thus, the above result can be reinterpreted for S-polynomials as follows: given polynomials $p_i, p_j, p_k \in \mathbb{Z}_m[\boldsymbol{x}]$ such that $\mathsf{lt}_\prec(p_i)$ divides $\mathsf{lcm}(\mathsf{lt}_\prec(p_j), \mathsf{lt}_\prec(p_k))$ then the S-polynomial $\mathsf{S}_\prec(p_i, p_k)$ can be dropped (from S in Fig. 2) if $\mathsf{S}_\prec(p_i, p_k)$ and $\mathsf{S}_\prec(p_i, p_j)$ are (eventually) computed. Although this rule mirrors the triple criteria of [13], modular polynomials offer additional redundancy rules. Together these rules reduce the running time from hours on the above 4-bit example to 510 ms on a 2.5 GHz 16 GB Macbook.

Our implementation is 9293 LOC of Scala3 and is stratified in 3 layers: the worklist-driven fixpoint engine, the domain operations and the underlying Gröbner basis solver. For bit-widths of 8, 12 and 16, the running times are 1,398 ms 5,894 ms and 54,019 ms respectively (though the actual target for our work is AVR micro-controller code which is merely 8-bit). To scale to these higher bit-widths, it is necessary to reduce the numbers of terms in each polynomial. Rather surprisingly, this can be achieved on-the-fly as a polynomial is generated term-by-term rather than as a post-processing step which is applied to some (huge) polynomial derived from an arithmetic operation. To illustrate, consider summing two polynomials q_1 and q_2 where $q_1 = t_1 + r_1$ and $q_2 = t_2 + r_2$ and t_1 and t_2 are leading terms with the same powers. The term $t = t_1 + t_2$ is computed then reduced (whenever possible) with a null polynomial whose leading term divides t to give a simplified polynomial q. Then we apply the same tactic to sum $q + r_1$ to give a polynomial s and apply the same tactic yet again to sum $s + r_2$. A null polynomial whose leading terms divides t can be found directly, provided one exists, without resorting to search or even a lookup table. In fact, given t, the null can be found simply by multiplying terms together [32, definition 9]. For example, if $\omega = 4$ and $t = 3x^2y^3$ then the null polynomial $3x^2y^3 + 7x^2y^2 + 6x^2y + 13xy^3 + 9xy^2 + 10xy$ is found by expanding $3x(x-1)y(y-1)(y-2)$ so that $q = 9x^2y^2 + 10x^2y + 3xy^3 + 7xy^2 + 6xy$. The degree of the leading term of $q_1 + q_2$ is then reduced from x^2y^3 to x^2y^2. Applying this tactic repeatedly keeps the number of terms small in all arithmetic operations. (The tactic is not mentioned in [2] but appears to be a dynamic version of

the technique used in [33] that applies null (vanishing) polynomials to statically bound the representation of polynomials.)

7 Related Work

Momentum for migrating abstract domains from idealised arithmetic to machine arithmetic is growing [10–12,21,26,27], driven by the desire to soundly model program behaviour and low-level code. Some of these domains [10,21,26,27] satisfy the ascending chain condition, which is key to computing best transformers [28] though, to our knowledge, this observation has not been previously made.

Early approaches to deriving (non-modular) polynomial invariants employed forwards [29] and backwards [25] abstract interpretation over domains of polynomial ideals. In the former case, termination of the analysis requires a widening operator to remove polynomials of high degree, since polynomial ideals over $\mathbb{Q}$ do not satisfy the ascending chain condition. In the latter case, the analysis is primed with a template polynomial of bounded degree; linear systems are then solved to find assignments to the (symbolic) coefficients of the templates, which yield the polynomial invariants. An alternative approach [31], also employing template polynomials, directly encodes the conditions for a given template polynomial to be an invariant as a parametric linear system, which can then be solved with suitable methods [6]. None of these analyses is complete and [25] concludes, "It is a challenging open problem whether or not the set of all polynomial relations can be computed not just ones of some given form".

This challenge [25] has motivated subsequent work [17,18,23,30], which restrict the form of programs that can be analysed, either to those containing only simple loops [30], P-solvable loops [23] or affine programs [17] (where a variable is assigned to an affine expression). State-of-the-art in computing all polynomial relations focuses on affine programs [17] where the problem is reduced to that of computing the Zariski closure of the semigroup generated by a finite set of rational square matrices. However, it is not clear how this approach extends to general polynomial assignments, particularly those in a modular setting.

A more promising line of enquiry in this vein [33] seeks to adapt backwards abstract interpretation to inferring non-modular polynomial invariants [25] to a modular setting. The insight is that it is possible to bound the degree of the template polynomial without losing precision, by exploiting the fact that any modular polynomial is semantically equivalent to one with degree at most $1.5(\omega + d)$. Building on this bound, the analysis of [33, pp. 311] seeks to infer all polynomial invariants for programs consisting of polynomial and non-deterministic assignments, non-deterministic branching and polynomial disequality guards. For disequality guards, the weakest precondition transformer is defined as $[\![p \neq 0]\!]^T q = \{\, pq \,\}$ [33, pp. 306]. The subtlety of the modular setting is that the pre-condition pq can vanish, compromising soundness. To illustrate, put $p = 2x$ and $q = 128x$ in $\mathbb{Z}_{256}[x]$. Then $pq = (2x)(128x) = 256x^2 = 0$, which holds vacuously. Now observe that the assignment $x = 1$, which satisfies 0, passes the disequality guard $2x \neq 0$ but violates q. Thus the weakest precondition transformer is actually unsound. This not only illustrates the delicacy of

modular reasoning, but suggests that the ability to reason about disequalities, even imprecisely, is a key advantage of the present work.

By design, the analysis [33] does not support equality guards, as these are not readily handled [33, pp. 301] by weakest precondition transformers. However, handling equalities is sometimes necessary, as demonstrated by the worked example, where $x - y = 0$ is required for inferring $ax - 1 = 0$ at program exit. Interestingly, the analysis [33] does not rely on Gröbner basis computations, but rather only exploits reduction and properties of null polynomials to detect fixed points. Although this finesses the need for Gröbner bases, it is not clear how they can be avoided when computing join in forwards analysis.

8 Conclusions

Working over modular integers is not merely more realistic, but reshapes the domain operations which can and need be applied. Widening is unnecessary since modular integers induce a domain of polynomial invariants which satisfies the ascending chain condition. Negative polynomial guards can be supported by partitioning the solution set of a polynomial disequality into sets of integers whose least bits equal a power of two. MPAD extends the scope of invariant discovery as demonstrated on an algorithm for calculating a multiplicative inverse.

Acknowledgements. We thank the anonymous reviewers for their insightful comments and Helmut Seidl for kindly checking the vanishing precondition example. This work was supported, in part, by EPSRC grant EP/T014512/1.

References

1. Adams, W., Loustaunau, P.: An Introduction to Gröbner Bases. American Mathematical Society (1994)
2. Brickenstein, M., Dreyer, A., Greuel, G., Wedler, M., Wienand, O.: New developments in the theory of Gröbner bases and applications to formal verification. J. Pure Appl. Algebra **213**, 1612–1635 (2009)
3. Buchberger, B.: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. J. Symb. Comput. **41**, 475–511 (2006)
4. Cacher, D., Jensen, T., Jobin, A., Kirchner, F.: Inference of polynomial invariants for imperative programs: a farewell to Gröbner bases. Sci. Comput. Program. **93**, 89–109 (2014)
5. Codish, M., Mulkers, A., Bruynooghe, M., Garcia De La Banda, M., Hermenegildo, M.: Improving abstract interpretations by combining domains. Trans. Program. Lang. Syst. **17**(1), 28–44 (1995)
6. Collins, G., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. J. Symb. Comput. **12**, 299–328 (1991)
7. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Principles of Programming Languages, pp. 238–252. ACM Press (1977)

8. de Oliveira, S., Bensalem, S., Prevosto, V.: Polynomial invariants by linear algebra. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 479–494. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_30
9. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: Sifakis, J. (ed.) CAV 1989. LNCS, vol. 407, pp. 197–212. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52148-8_17
10. Elder, M., Lim, J., Sharma, T., Andersen, T., Reps, T.: Abstract domains of affine relations. Trans. Program. Lang. Syst. **36**(4), 1–73 (2014)
11. Gange, G., Navas, J., Schachte, P., Søndergaard, H., Stuckey, P.: Interval analysis and machine arithmetic: why signedness ignorance is bliss. Trans. Program. Lang. Syst. **37**(1), 1–35 (2014)
12. Gange, G., Søndergaard, H., Stuckey, P.J., Schachte, P.: Solving difference constraints over modular arithmetic. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 215–230. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_15
13. Gebauer, R., Möller, H.: On an installation of Buchberger's algorithm. J. Symb. Comput. **6**(2/3), 275–286 (1988)
14. Greuel, G.M., Seelisch, F., Wienand, O.: The Gröbner basis of the ideal of vanishing polynomials. J. Symb. Comput. **46**(5), 561–570 (2011)
15. Harrison, W.: Compiler analysis of the value ranges for variables. IEEE Trans. Softw. Eng. **3**(3), 243–250 (1977)
16. Hilbert, D.: Über die Theorie der Algebraischen Formen. Math. Ann. **36**(4), 473–534 (1890)
17. Hrushovski, E., Ouaknine, J., Pouly, A., Worrell, J.: Polynomial invariants for affine programs. In: Logic in Computer Science, pp. 530–539. ACM Press (2018)
18. Humenberger, A., Jaroschek, M., Kovács, L.: Invariant generation for multi-path loops with polynomial assignments. In: Dillig, I., Palsberg, J. (eds.) VMCAI 2018. LNCS, vol. 10747, pp. 226–246. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73721-8_11
19. Huynh, D.: A super-exponential lower bound for Gröbner bases and Church-Rosser commutative Thue systems. Inf. Control **68**(1–3), 196–206 (1986)
20. Karr, M.: Affine relationships among variables of a program. Acta Informatica **6**(2), 133–151 (1976)
21. King, A., Søndergaard, H.: Inferring congruence equations using SAT. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 281–293. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70545-1_26
22. Kovács, L.: Reasoning algebraically about P-solvable loops. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 249–264. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_18
23. Kovács, L.: A complete invariant generation approach for P-solvable loops. In: Pnueli, A., Virbitskaite, I., Voronkov, A. (eds.) PSI 2009. LNCS, vol. 5947, pp. 242–256. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11486-1_21
24. Miné, A.: The octagon abstract domain. High. Order Symb. Comput. **19**(1), 31–100 (2006)
25. Müller-Olm, M., Seidl, H.: Computing polynomial program invariants. Inf. Process. Lett. **91**, 233–244 (2004)
26. Müller-Olm, M., Seidl, H.: Analysis of modular arithmetic. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 46–60. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31987-0_5

27. Müller-Olm, M., Seidl, H.: Analysis of modular arithmetic. Trans. Program. Lang. Syst. **29**(5), 1–26 (2007)
28. Reps, T., Sagiv, M., Yorsh, G.: Symbolic implementation of the best transformer. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 252–266. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24622-0_21
29. Rodríguez-Carbonell, E., Kapur, D.: An abstract interpretation approach for automatic generation of polynomial invariants. In: Giacobazzi, R. (ed.) SAS 2004. LNCS, vol. 3148, pp. 280–295. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27864-1_21
30. Rodríguez-Carbonell, E., Kapur, D.: Generating all polynomial invariants in simple loops. J. Symb. Comput. **42**, 443–476 (2007)
31. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Non-linear loop invariant generation using Gröbner bases. In: Principles of Programming Languages, pp. 318–329. ACM Press (2004)
32. Seed, T.: Program verification using polynomials over modular arithmetic. Ph.D. thesis, University of Kent (2021). https://doi.org/10.22024/UniKent/01.02.90261. https://kar.kent.ac.uk/90261/
33. Seidl, H., Flexeder, A., Petter, M.: Analysing all polynomial equations in $\mathbb{Z}_{2^w}$. In: Alpuente, M., Vidal, G. (eds.) SAS 2008. LNCS, vol. 5079, pp. 299–314. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69166-2_20
34. Warren, H.: Hacker's Delight. Addison-Wesley, Boston (2012)