Two Variables per Linear Inequality as an Abstract Domain

Axel Simon[1] Andy King[1] Jacob M. Howe[2]

[1]Computing Laboratory,
University of Kent, Canterbury, UK.
{a.m.king, a.simon}@ukc.ac.uk

[2]Department of Computing,
City University, London, UK.
jacob@soi.city.ac.uk

Abstract. This paper explores the spatial domain of sets of inequalities where each inequality contains at most two variables – a domain that is richer than intervals and more tractable than general polyhedra. We present a complete suite of efficient domain operations for linear systems with two variables per inequality with unrestricted coefficients. We exploit a tactic in which a system of inequalities with at most two variables per inequality is decomposed into a series of projections – one for each two dimensional plane. The decomposition enables all domain operations required for abstract interpretation to be expressed in terms of the two dimensional case. The resulting operations are efficient and include a novel planar convex hull algorithm. Empirical evidence suggests that widening can be applied effectively, ensuring tractability.

1 Introduction

The value of spatial domains such as intervals [13], affine spaces [19] and polyhedra [8] has been recognized since the early days of program analysis. One reoccurring theme in program analysis is the trade-off between precision of the domain and the tractability of the domain operations. In this regard, the polyhedral sub-domain that consists of sets of linear inequalities where each inequality contains at most two variables has recently attracted attention [26, 27, 33, 35]. In fact, because of its tractability, this class of linear inequalities has recently been proposed for constraint logic programming [15, 18]. This paper adapts this work to the requirements of program optimization and program development by equipping this domain with the operations needed for abstract interpretation. Two variable inequality domains have already proven useful in areas as diverse as program verification [29, 34], model checking of timed automata [22, 28], parallelization [2], locating security vulnerabilities [36], detecting memory leaks [33] and verifying program termination in logic programming [24]. Thus the applicability of the domain extends beyond logic programming [4, 17] to other analysis problems in verification and program development.

The work of Miné [26] represents the state-of-the-art for program analysis with domains of inequalities restricted to two variables. He uses the so-called Octagon domain [26] where inequalities have unit coefficients of -1, 0 or +1. A difference-bound matrix (DBM) representation is employed that uses a $2d \times 2d$

matrix to encode a system of inequalities, S say, over d variables (the dimension). One key idea in this work is that of closure. Closure strengthens the inequalities of S (represented as a DBM) to obtain a new system S' (also represented as a DBM). For example, if $x+y \leq c' \in S'$, then $c' \leq c$ whenever S implies $x+y \leq c$. Thus applying closure maximally tightens each inequality, possibly introducing new inequalities. Projection, entailment and join apply closure as a preprocessing step both to preserve precision and simplify the domain operations themselves. For example, the join of two inequalities with identical coefficients, say $x-y \leq c_1$ and $x - y \leq c_2$, is simply $x - y \leq \max(c_1, c_2)$. Closure enables this simple join to be lifted point-wise to systems of inequalities. Since most domain operations require one or both of their arguments to be closed, these operations inherit the $O(d^3)$ complexity of the DBM closure operation. In this paper, we show how closure is also the key concept to tackle the two variable per inequality domain with unrestricted coefficients. Henceforth, our closure operator is referred to as completion to distinguish it from topological closure.

This paper draws together a number of strands from the verification, analysis and constraints literature to make the following novel contributions:

- We show that a polynomial completion algorithm which makes explicit all the two-dimensional projections of a system of (unrestricted) two variable inequalities enables each domain operation to be computed in polynomial time. Incredibly, such a completion operator already exists and is embedded into the satisfiability algorithm of Nelson [29].
- We explain how classic $O(m \log m)$ convex hull algorithms for sets of m planar points, such as [11], can be adapted to compute the join efficiently. The crucial point is that completion enables join to be computed point-wise on each two-dimensional projection which necessarily describes a planar object. Surprisingly little literature addresses how to efficiently compute convex hull of planar polyhedra (without the full complexity of the standard d-dimensional algorithm [6, 23]) and as far as we are aware, our convex hull algorithm is unique (see [32] for a recent survey). Projection and entailment operators are also detailed.
- We also address scalability and present empirical evidence that the number of inequalities in each two-dimensional projection is small. This suggests a natural widening: limit the number of inequalities in each projection by a constant. This trivial widening obtains an $O(d^2)$ representation, like DBMs, without enforcing the requirement that coefficients are $-1, 0$ or $+1$. Note that in contrast to DBMs, our representation is dense – space is only required for those inequalities actually occurring in the system. The widening also causes completion to collapse to an $O(d^3(\log d)^2)$ operation which is competitive with the $O(d^3)$ DBM approach, taking into consideration the extra expressiveness.
- We also argue that the domain operations themselves are conceptually simple, straightforward to code and therefore more likely to be implemented correctly.

To summarize, we remove a serious limitation of the Octagon domain – that the coefficients must be unitary – without compromising tractability. Applications that employ the Octagon domain or related weaker domains [22, 28, 33] will therefore directly benefit from this work.

The paper is structured as follows. Section 2 presents the abstract domain. Section 3 explains how Nelson's satisfiability algorithm [29] can be adapted to complete a system. The next three sections explain how completion provides the basis for the domain operations. Section 7 presents empirical evidence for the practicality of the domain. The future and related work sections conclude.

2 Abstract domain

To specify the domain algorithms and argue their correctness, we start the exposition by detailing some theoretical properties of polyhedral domains.

2.1 Convex hull and closure

An ϵ-ball around $\boldsymbol{y} \in \mathbb{R}^n$ is defined as $B_\epsilon(\boldsymbol{y}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \sum_{i=1}^{n}(x_i - y_i)^2 < \epsilon\}$. A set $S \subseteq \mathbb{R}^n$ is open if, given any $y \in S$, there exists $\epsilon > 0$ such that $B_\epsilon(y) \subseteq S$. A set $S \subseteq \mathbb{R}^n$ is closed iff $\mathbb{R}^n \setminus S$ is open. Note that if $S_i \subseteq \mathbb{R}^n$ is closed for each member of an index set $i \in I$ then $\cap\{S_i \mid i \in I\}$ is also closed. The (topological) closure of $S \in \mathbb{R}^n$ is defined $cl(S) = \cap\{S' \subseteq \mathbb{R}^n \mid S \subseteq S' \wedge S' \text{ is closed}\}$. The convex hull of $S \in \mathbb{R}^n$ is defined $conv(S) = \{\lambda\boldsymbol{x}+(1-\lambda)\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{y} \in S \wedge 0 \leq \lambda \leq 1\}$.

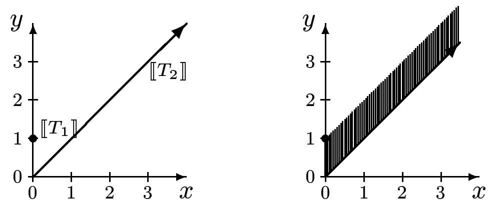
2.2 Two-variables per inequality domain

Let X denote the finite set of variables $\{x_1, \ldots, x_n\}$ so that X is ordered lexicographically by $x_i \prec x_j$ iff $i < j$. Let Lin_X denote the set of (possibly rearranged) linear inequalities of the form $ax_i + bx_j \leq c$ where $a, b, c \in \mathbb{R}$. Let Two_X denote the set of all finite subsets of Lin_X. Note that although each set $T \in Two_X$ is finite, Two_X is not finite. Syntactic sugar of the form $x \leq y$ is used instead of $(+1)x + (-1)y \leq 0 \in Lin_X$ as well as $by + ax \leq c$ instead of $ax + by \leq c$.

Definition 1. The mapping $[\![.]\!] : Lin_X \to \mathbb{R}^n$ is defined: $[\![ax_i + bx_j \leq c]\!] = \{\langle y_1, \ldots, y_n\rangle \in \mathbb{R}^n \mid ay_i + by_j \leq c\}$ and the mapping $[\![.]\!] : Two_X \to \mathbb{R}^n$ is defined $[\![T]\!] = \cap\{[\![t]\!] \mid t \in T\}$.

For brevity, let $t^=$ represent the boundary of a given half-space, that is, define $t^= = \{ax_i + bx_j \leq c, -ax_i - bx_j \leq -c\}$ when $t \equiv ax_i + bx_j \leq c$. Two_X is ordered by entailment, that is, $T_1 \models T_2$ iff $[\![T_1]\!] \subseteq [\![T_2]\!]$. Equivalence on Two_X is defined $T_1 \equiv T_2$ iff $T_1 \models T_2$ and $T_2 \models T_1$. Moreover $T \models t$ iff $T \models \{t\}$ and $t_1 \equiv t_2$ iff $\{t_1\} \equiv \{t_2\}$. Let $Two_X^{\equiv} = Two_X/\equiv$. $Two_X^{\equiv}$ inherits entailment $\models$ from Two_X. In fact $\langle Two_X^{\equiv}, \models, \sqcap, \sqcup\rangle$ is a lattice (rather than a complete lattice) with $[T_1]_\equiv \sqcap [T_2]_\equiv = [T_1 \cup T_2]_\equiv$ and $[T_1]_\equiv \sqcup [T_2]_\equiv = [T]_\equiv$ where $[\![T]\!] = cl(conv([\![T_1]\!] \cup [\![T_2]\!]))$. Note that in general $conv([\![T_1]\!] \cup [\![T_2]\!])$ is not closed and therefore cannot be described by a system of non-strict linear inequalities as is illustrated below.

Example 1. Let $X = \{x, y\}$, $T_1 = \{x \leq 0, -x \leq 0, y \leq 1, -y \leq -1\}$ and $T_2 = \{-x \leq 0, x - y \leq 0, y - x \leq 0\}$ so that $[\![T_1]\!] = \{\langle 0, 1\rangle\}$ and $[\![T_2]\!] = \{\langle x, y\rangle \mid 0 \leq x \wedge x = y\}$. Then $conv([\![T_1]\!] \cup [\![T_2]\!])$ includes the point $\langle 0, 1\rangle$ but not the ray $\{\langle x, y\rangle \mid 0 \leq x \wedge x + 1 = y\}$ and hence is not closed.



The domain Two_X is a generic abstract domain that is not limited to a specific application. No concretization map is defined in this paper since such a map is specific to an application. However, if an application used the concretization map $\gamma(T) = [\![T]\!]$ then no abstraction map $\alpha : \wp(\mathbb{R}^n) \rightarrow Two_X$ would exist since there is no best abstraction e.g. for the set $\{\langle x, y\rangle \mid x^2 + y^2 \leq 1\}$. The problem stems from the fact that Two_X can contain an arbitrarily large number of inequalities. This contrasts with the Octagon domain where each planar object will be described by at most eight inequalities.

We will augment $\langle Two_X^{\equiv}, \models, \sqcap, \sqcup\rangle$ with projection $\exists$ and widening to accommodate the needs of abstract interpretation.

Definition 2. Projection operator $\exists_{x_i} : Two_X^{\equiv} \rightarrow Two_X^{\equiv}$ is defined $\exists_{x_i}([T_1]_{\equiv}) = [T_2]_{\equiv}$ where $[\![T_2]\!] = \{\langle y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_n\rangle \mid y \in \mathbb{R} \wedge \langle y_1, \ldots, y_n\rangle \in [\![T_1]\!]\}$.

Projection can be calculated using Fourier-Motzkin variable elimination and from this it follows that $T_2 \in Two_X$ if $T_1 \in Two_X$.

2.3 Complete form for the two-variables per inequality domain

The complete form for the two-variables per inequality domain is defined in terms of those variables that occur in a set of inequalities.

Definition 3. The mapping $var : Lin_X \rightarrow \wp(X)$ is defined:

$$var(ax + by \leq c) = \begin{cases} \emptyset & \text{if } a = b = 0 \\ \{y\} & \text{if } a = 0 \\ \{x\} & \text{if } b = 0 \\ \{x, y\} & \text{otherwise} \end{cases}$$

The mapping var captures those variables with non-zero coefficients. Observe that $var(t_1) = var(t_2)$ if $t_1 \equiv t_2$. In contrast, note that $var(0u + 0v \leq 1) = \emptyset = var(0x + 0y \leq -1)$. If $T \in Two_X$ then let $var(T) = \cup\{var(t) \mid t \in T\}$.

Definition 4. Let $Y \subseteq X$. The restriction operator π_Y is defined:

$$\pi_Y(T) = \{t \in T \mid var(t) \subseteq Y\}$$

Definition 5. The set of *complete* finite subsets of Lin_X is defined:

$$Two'_X = \{T \in Two_X \mid \forall t \in Lin_X \,.\, T \models t \;\Rightarrow\; \pi_{var(t)}(T) \models t\}$$

Proposition 1. Suppose $T \in Two_X$. Then there exists $T' \in Two'_X$ such that $T \subseteq T'$ and $T \equiv T'$.

Proof. Define $[T_{x,y}]_\equiv = \exists_{X\setminus\{x,y\}}([T]_\equiv)$ for all $x, y \in X$ and $T' = T \cup \bigcup_{x,y\in X} T_{x,y}$. Since each $T_{x,y}$ is finite, T' is finite, hence $T' \in Two'_X$. By the definition of $\exists$, $T \models T_{x,y}$, hence $T \cup T_{x,y} \equiv T$ for all $x, y \in X$, thus $T \equiv T'$. Moreover $T \subseteq T'$. ▮

Corollary 1. $Two^{\equiv}_X = Two'_X/\equiv$.

2.4 Ordering the two-variables per inequality domain

Let $Y = \{x, y\} \subseteq X$ such that $x \prec y$ and consider $T = \{t_1, \ldots, t_n\} \in Two_Y$. Each t_i defines a half-space in the Y plane and therefore T can be ordered by the orientation of the half-spaces as follows:

Definition 6. The (partial) mapping $\theta : Lin_Y \to [0, 2\pi)$ is defined such that $\theta(ax + by \leq c) = \psi$ where $\cos(\psi) = -b/\sqrt{a^2 + b^2}$ and $\sin(\psi) = a/\sqrt{a^2 + b^2}$.

The mapping θ actually returns the anti-clockwise angle which the half-space $\{\langle x, y\rangle \mid y \geq 0\}$ has to be turned through to coincide with $\{\langle x, y\rangle \mid ax + by \leq 0\}$.

2.5 Entailment between three inequalities

This section demonstrates how entailment checks of the form $\{t_1\} \models t$ and $\{t_1, t_2\} \models t$ can be computed in constant time. The following proposition explains how this check reduces to applying the Cramer rule for the three inequality case and simple scaling for the two inequality case.

Proposition 2. Let $t_i \equiv a_i x + b_i y \leq c_i$ for $i = 1, 2$ and $t \equiv ax + by \leq c$. Then

$$\{t_1\} \models t \iff \begin{cases} \mathit{false} & \text{if } a_1 b - a b_1 \neq 0 \\ \mathit{false} & \text{else if } a_1 a < 0 \vee b_1 b < 0 \\ (a/a_1)c_1 \leq c & \text{else if } a_1 \neq 0 \\ (b/b_1)c_1 \leq c & \text{else if } b_1 \neq 0 \\ c_1 < 0 \vee (c \geq 0 \wedge a = 0 \wedge b = 0) & \text{otherwise} \end{cases}$$

$$\{t_1, t_2\} \models t \iff \begin{cases} \{t_1\} \models t \vee \{t_2\} \models t & \text{if } d = a_1 b_2 - a_2 b_1 = 0 \\ \mathit{false} & \text{else if } \lambda_1 = (a b_2 - a_2 b)/d < 0 \\ \mathit{false} & \text{else if } \lambda_2 = (a_1 b - a b_1)/d < 0 \\ \lambda_1 c_1 + \lambda_2 c_2 \leq c & \text{otherwise.} \end{cases}$$

If the inequalities t_1 and t differ in slope, then the determinant of their coefficients is non-zero and they cannot entail each other. Suppose now that the determinant is zero. Observe that the two inequalities have opposing feasible spaces whenever a_1 and a or b_1 and b have opposite signs. In this case t_1 cannot entail t. If t_1 has a non-zero coefficient, then entailment reduces to a simple comparison between the constants of the inequalities, suitably scaled. The fifth case matches the pathological situation of tautologous and unsatisfiable inequalities.

The entailment between three inequalities reduces to the former case if t_1 and t_2 have equal slope (the determinant is zero). Otherwise an inequality is constructed which has the same slope as t and which passes through the intersection point $[\![t_1^=]\!] \cap [\![t_2^=]\!]$ using the Cramer rule. Again, a comparison of the constants determines the entailment relationship. If either λ_1 or λ_2 is negative, the feasible space of the combination of t_1 and t_2 will oppose that of t, thus $\{t_1, t_2\}$ cannot entail t.

3 Completion: A variant of Nelson's satisfiability algorithm

In this section we show how to complete a system of inequalities. This operation corresponds to the closure operation of Miné. We follow the approach that Nelson used for checking satisfiability [29]. One key concept in his algorithm is the notion of a filter that is formalized below.

Definition 7. Let $Y = \{x, y\} \subseteq X$. The mapping $\mathit{filter}_Y : \mathit{Two}_Y \to \mathit{Two}_Y$ is defined such that:

1. $\mathit{filter}_Y(T) \subseteq T$
2. $\mathit{filter}_Y(T) \equiv T$
3. for all $T' \subseteq T$ and $T' \equiv T$, $|\mathit{filter}_Y(T)| \leq |T'|$.

The role of filter_Y is to remove redundant elements from a set of inequalities over the variables Y. If the inequalities are ordered by angle, redundancy removal can be done surprisingly efficiently as illustrated in Fig. 1. The function *filter* returns a single contradictory inequality if the completed system S is unsatisfiable, and otherwise removes tautologies before sorting the inequalities. The loop then iterates over the inequalities once in an anti-clockwise fashion. It terminates when no more redundant inequalities can be found, that is, when (1) the whole set of inequalities has been traversed once (flag f is true) and (2) the inequalities with the largest and smallest angle are both non-redundant. Since the entailment check between three inequalities can be performed in constant time, the algorithm is linear. Note that different subsets of the input can be minimal. This occurs, for example, when the system is unsatisfiable. Then filter_Y returns one of these subsets.

The map filter_Y lifts to arbitrary systems of two-variable inequalities as follows:

Definition 8. The mapping $\mathit{filter} : \mathit{Two}_X \to \mathit{Two}_X$ is defined:

$$\mathit{filter}(T) = \bigcup\{\mathit{filter}_Y(\pi_Y(T)) \mid Y \subseteq X \wedge |Y| = 2\}$$

```
function filter_{x,y}(S ∈ Two_X) begin
    if ∃s ∈ S . s ≡ 0x + 0y ≤ −1 then return {s};
    T := {s ∈ S | s ≢ 0x + 0y ≤ 1};
    let T = {t_1, ..., t_m} such that θ(t_1) ≤ θ(t_2) ≤ ... ≤ θ(t_m);
    f := false;
    loop
        let {t_c, t_n, ..., t_l} = T; if |T| > 1 ∧ {t_n, t_l} ⊨ t_c then T := {t_n, ..., t_l}; else begin
            if θ(t_c) ≤ θ(t_l) ∧ f then return T;
            if θ(t_c) ≤ θ(t_l) then f := true;
            T := {t_l, t_c, t_n, ...};
        end;
    end;
end
```

Fig. 1. Algorithm for redundancy removal

The second key idea of Nelson is the *result* map that makes explicit those inequalities that are indirectly expressed by the system. The basic step is to generate all possible combinations of pairs of inequalities by eliminating their common variable.

Definition 9. The resultants map $result : Two_X \to Two_X$ is defined by:

$$result(T) = \left\{ aez - dby \le af - dc \;\middle|\; \begin{array}{l} t_1, t_2 \in T \quad\quad\quad \wedge \\ t_1 \equiv ax + by \le c \wedge \\ t_2 \equiv dx + ez \le f \wedge \\ a > 0 \wedge d < 0 \end{array} \right\}$$

The following example demonstrates how *result* works on a chain of dependent variables:

Example 2. Let $T_0 = \{x_0 \le x_1, x_1 \le x_2, x_2 \le x_3, x_3 \le x_4\}$. We calculate $T_1 = result(T_0)$ and $T_2 = result(T_0 \cup T_1)$.

$$\begin{aligned} result(T_0) &= \{x_0 \le x_2, x_1 \le x_3, x_2 \le x_4\} \\ result(T_0 \cup T_1) &= T_1 \cup \{x_0 \le x_3, x_0 \le x_4, x_1 \le x_4\} \end{aligned}$$

Note that $T_3 = \bigcup_{i=0}^{2} T_i$ is a fixpoint in $T_3 = result(T_3)$.

An important property of $T \cup result(T)$ is the way it halves the number of variables required to entail a given inequality t. Specifically, suppose $T \models t$. Then there exists $T' \subseteq T \cup result(T)$ such that $T' \models t$ and T' contains no more than half the variables of T. Lemma 1 formalizes this and is basically a reformulation of Lemma 1b of [29].

Lemma 1. Let $T \in Two_X$ and $t \in Lin_X$ such that $T \models t$. Then there exists $Y \subseteq X$ such that $|Y| \le \lfloor |var(T)|/2 \rfloor + 1$ and $\pi_Y(T \cup result(T)) \models t$.

Lemma 1 suggests the following iterative algorithm for calculating completion that takes (approximately) $\log_2(|var(T)|)$ steps. Theorem 1 asserts its correctness.

Definition 10. The mapping $complete : Two_X \to Two_X$ is defined:

$$complete(T_0) = T_{\lceil \log_2(|var(T_0)|-1) \rceil} \text{ where } T_{i+1} = filter(T_i \cup result(T_i))$$

Theorem 1. *$complete(T) \equiv T$ and $complete(T) \in Two'_X$ for all $T \in Two_X$.*

Proof. Let $f : \mathbb{N} \to \mathbb{N}$ where $f(n) = \lfloor n/2 \rfloor + 1$. The following table details $m \in \mathbb{N}$ for which $f^m(n) \leq 2$. Observe that $f^{\lceil \log_2(n-1) \rceil}(n) \leq 2$.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
m	0	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	5	...

Observe that $T \equiv T \cup result(T) \equiv filter(T \cup result(T))$ and by induction $T \equiv complete(T)$. Let $t \in Lin_X$ such that $complete(T) \models t$. Then $T \models t$. Let $T_0 = T$ and $T_{i+1} = filter(T_i \cup result(T_i))$. By induction and by Lemma 1, there exists $Y_i \subseteq var(T)$ such that $\pi_{Y_i}(T_i) \models t$ and $|Y_i| \leq f^i(|var(T)|)$. Therefore $|Y_{\lceil \log_2(|var(T)|-1) \rceil}| \leq 2$, hence $\pi_{var(t)}(complete(T)) \models t$ as required. ∎

Note that applying an additional completion step makes explicit all inequalities over one variable. Furthermore, applying it once more creates tautologous and contradictory inequalities. Applying these two additional completion steps enables filter to detect unsatisfability without employing any extra machinery.

Example 3. To illustrate how unsatisfiability is detected consider the system $T_0 = \{-x+y \leq -1, -2x-3y \leq -6, 4x-2y \leq -4\}$. The system is complete but two more completion steps are necessary to detect unsatisfiability. The calculation $T_1 = filter(T_0 \cup result(T_0)) = T_0 \cup \{-y \leq -2, -5x \leq -9, x \leq -3\}$ makes all inequalities over one variable explicit. Unsatisfiability becomes explicit when calculating $0 \leq -24 \in result(T_1)$. Finally $filter(result(T_1)) = \{0 \leq -24\}$ collapses the system to a single unsatisfiable constraint.

3.1 Complexity of the *complete* operation

Nelson shows that his satisfiability algorithm is polynomial in the number of input inequalities [29]. For comparison with the DBM approach, consider the complexity of $filter(T_i \cup result(T_i))$ where $d = |var(T_i)|$ and $k = \max\{|\pi_Y(T_i)| \mid i \in [0, \lceil \log_2(|var(T)| - 1) \rceil] \wedge Y = \{x, y\} \subseteq var(T_i)\}$. Since each T_i may have $d(d-1)/2$ restrictions, a linear pass over $O(kd^2)$ inequalities is sufficient to partition the set of inequalities into d sets, one for each variable. Each set has at most $O(kd)$ elements, so calculating the resultants for each set is $O(k^2d^2)$, hence calculating all the resultants is $O(k^2d^3)$. The complexity of applying the linear *filter* is in $O(kd^2 + k^2d^3) = O(k^2d^3)$ which with sorting requires $O(k^2d^3\log(k^2d^3)) = O(k^2d^3(\log(k)+\log(d)))$ time. The *complete* operation runs *result* $O(\log d)$ times which leads to an overall running time of $O(k^2d^3\log(d)(\log(k) + \log(d)))$. In Section 7 we show that k is typically small and therefore can be limited by a constant with hardly any loss of expressiveness. This collapses the bound to $O(d^3(\log(d))^2)$ which is only slightly worse than the $O(d^3)$ closure of Miné [26].

3.2 Satisfiability and the *complete* operation

Nelson [29] originally devised this completion operation in order to construct a polynomial test for satisfiability. The following proposition explains how non-satisfiability can be observed after (and even during) the completion calculation. Specifically, the proposition asserts that non-satisfiability always manifests itself in the existence of at least one contradictory inequality.

Proposition 3. Let $T' \in Two'_X$. Then $[\![T']\!] = \emptyset$ iff $[\![\pi_\emptyset(T')]\!] = \emptyset$.

Proof. Let $T' \in Two'_X$. Suppose $[\![T']\!] = \emptyset$. Then $T' \models 0x + 0y \leq -1$. Since $var(0x+0y \leq -1) = \emptyset$, hence $\pi_\emptyset(T') \models 0x+0y \leq -1$ and therefore $[\![\pi_\emptyset(T')]\!] = \emptyset$. Since $\pi_\emptyset(T') \subseteq T'$ the converse follows.

4 Join: Planar convex hull on each projection

Computing the join corresponds to calculating the convex hull for polyhedra which is surprisingly subtle. The standard approach for arbitrary d-dimensional polyhedra involves applying the Chernikova [6] algorithm (or a variant [23]) to construct a vertices and rays representation which is potentially exponential [20]. By way of contrast, we show that convex hull for systems of two variables per inequality can be computed by a short polynomial algorithm.

The construction starts by reformulating the convex hull piece-wise in terms of each of its planar projections. Proposition 4 shows that this operation results in a complete system whenever its inputs are complete; equivalence with the fully dimensional convex hull operation is stated in Proposition 5.

Definition 11. The piece-wise convex hull $\curlyvee : {Two_X}^2 \to Two_X$ is defined $T_1 \curlyvee T_2 = \cup\{T_{x,y} \in Two_{\{x,y\}} \mid x, y \in X\}$ where $[\![T_{x,y}]\!] = cl(conv([\![\pi_{\{x,y\}}(T_1)]\!] \cup [\![\pi_{\{x,y\}}(T_2)]\!]))$.

Proposition 4. $T'_1 \curlyvee T'_2 \in Two'_X$ if $T'_1, T'_2 \in Two'_X$.

Proof. Let $t \in Lin_X$ such that $T'_1 \curlyvee T'_2 \models t$. Let $x, y \in X$ and let $[\![T_{x,y}]\!] = cl(conv([\![\pi_{\{x,y\}}(T'_1)]\!] \cup [\![\pi_{\{x,y\}}(T'_2)]\!]))$. Observe $\pi_{\{x,y\}}(T'_1) \models T_{x,y}$, therefore $T'_1 \models T'_1 \curlyvee T'_2$. Likewise $T'_2 \models T'_1 \curlyvee T'_2$, hence it follows that $T'_1 \models t$ and $T'_2 \models t$. Since $T'_1, T'_2 \in Two'_X$, $\pi_{var(t)}(T'_1) \models t$ and $\pi_{var(t)}(T'_2) \models t$, thus $[\![\pi_{var(t)}(T'_1)]\!] \subseteq [\![t]\!]$ and $[\![\pi_{var(t)}(T'_2)]\!] \subseteq [\![t]\!]$, hence $[\![\pi_{var(t)}(T'_2)]\!] \cup [\![\pi_{var(t)}(T'_2)]\!] \subseteq [\![t]\!]$. Therefore $[\![\pi_{var(t)}(T'_1 \curlyvee T'_2)]\!] = cl(conv([\![\pi_{var(t)}(T'_1)]\!] \cup [\![\pi_{var(t)}(T'_2)]\!])) \subseteq cl(conv([\![t]\!])) = [\![t]\!]$. Therefore $\pi_{var(t)}(T'_1 \curlyvee T'_2) \models t$ as required. ∎

Proposition 5. $[\![T'_1 \curlyvee T'_2]\!] = cl(conv([\![T'_1]\!] \cup [\![T'_2]\!]))$ if $T'_1, T'_2 \in Two'_X$.

Proof. Since $T'_1 \models T'_1 \curlyvee T'_2$ and $T'_2 \models T'_1 \curlyvee T'_2$, it follows that $cl(conv([\![T'_1]\!] \cup [\![T'_2]\!])) \subseteq [\![T'_1 \curlyvee T'_2]\!]$. Suppose there exists $\langle c_1, \ldots, c_n\rangle \in [\![T'_1 \curlyvee T'_2]\!]$ such that $\langle c_1, \ldots, c_n\rangle \notin [\![T']\!]$ where $[\![T']\!] = cl(conv([\![T'_1]\!] \cup [\![T'_2]\!]))$. Thus $\bigcup_{i=1}^n \{x_i \leq c_i, c_i \leq x_i\} \not\models T'$, hence there exists $ax_j + bx_k \leq c \equiv t \in T'$ with $\bigcup_{i=1}^n \{x_i \leq c_i, c_i \leq x_i\} \not\models ax_j + bx_k \leq c$.

```
function extreme(T ∈ Two_{x,y}) begin
    let T = {t_0, ..., t_{n-1}} such that θ(t_0) < θ(t_1) < ... < θ(t_{n-1});
    V := R := ∅;
    for i ∈ [0, n-1] do let t_i ≡ ax + by ≤ c in begin
        // are the intersection points of this inequality degenerated?
        d_pre := (θ(t_i) - θ(t_{i-1 mod n})) mod 2π ≥ π ∨ n = 1;
        d_post := (θ(t_{i+1 mod n}) - θ(t_i)) mod 2π ≥ π ∨ n = 1;
        if d_pre then R := R ∪ {⟨b/√(a²+b²), -a/√(a²+b²)⟩};
        if d_post then R := R ∪ {⟨-b/√(a²+b²), a/√(a²+b²)⟩};
                else V := V ∪ {v} where v ∈ ⟦t_i^=⟧ ∩ ⟦t_{(i+1) mod n}^=⟧;
        if d_pre ∧ d_post then begin
            if n = 1 then R := R ∪ {⟨-a/√(a²+b²), -b/√(a²+b²)⟩};
            V := V ∪ {v} where v ∈ ⟦t_i^=⟧
        end
    end
    return ⟨V, R⟩
end
```

Fig. 2. Calculating the points and rays of a planar polyhedron

But $T_1' \models T' \models t$ and $T_2' \models T' \models t$. Since $T_1' \in Two'_X$ and $T_2' \in Two'_X$, it follows that $\pi_{\{x_j,x_k\}}(T_1') \models t$ and $\pi_{\{x_j,x_k\}}(T_2') \models t$. Hence $T_1' \curlyvee T_2' \models t$, thus $\bigcup_{i=1}^{n}\{x_i \leq c_i, c_i \leq x_i\} \models T_1' \curlyvee T_2'$ but $\langle c_1, \ldots, c_n\rangle \notin [\![T_1' \curlyvee T_2']\!]$ which is a contradiction. ∎

Calculating the convex hull for a set of points in the plane has been studied extensively [32]. The convex hull of polytopes can be reduced to this problem by converting the polytopes into their vertex representation, calculating the convex hull of all vertices and converting back into the inequality representation. Although the generalization to planar polyhedra follows this three-step process, it is much more subtle and little literature has been written on this fundamental problem. Given a set of non-redundant inequalities, ordered by their orientation θ, the auxiliary function *extreme* in Figure 2 calculates a set of vertices and rays that represent the polyhedron. Rays are created when the angle between the current inequality t_i and the previous inequality is greater or equal to π (d_{pre} is true) and similarly for the next inequality (d_{post} is true). If both flags are true, we create an arbitrary point on the boundary of the halfspace of t_i to fix its representing rays in space. A pathological case arises when the polyhedron consists of a single halfspace ($n = 1$). In this case a third ray is created to indicate on which side the feasible space lies. Note that the maximum number of rays for each polyhedron is four, which occurs when T defines two facing halfspaces.

The main function *join* in Figure 3 uses *extreme* to compute the vertices and rays of each input polyhedron and catches the simple case of when both polyhedra consist of the same single point. Otherwise we calculate a square whose sides have length $2m$ which is centered on the origin and that contains all vertices

```
function join(T₁ ∈ Two_X, T₂ ∈ Two_X) begin
    if ∃t ∈ T₁ . t ≡ 0x + 0y ≤ −1 then return T₂;
    if ∃t ∈ T₂ . t ≡ 0x + 0y ≤ −1 then return T₁;
    // note: each T_i is non-redundant
    ⟨V₁, R₁⟩ := extreme(T₁);
    ⟨V₂, R₂⟩ := extreme(T₂);
    V := V₁ ∪ V₂; R := R₁ ∪ R₂; // Note: |R| ≤ 8
    if V = {⟨x₁, y₁⟩} ∧ R = ∅ then
        return {x ≤ x₁, −x ≤ −x₁, y ≤ y₁, −y ≤ −y₁};
    m := max{|x|, |y| | ⟨x, y⟩ ∈ V} + 1;
    //add a point along the ray, goes through x, y and is outside the box
    for ⟨x, y, a, b⟩ ∈ V₁ ∪ V₂ × R do V := V ∪ {⟨x + 2√2ma, y + 2√2mb⟩};
    {v₀, ..., v_{n−1}} := graham(V) such that v₀, ..., v_{n−1} are ordered anti-clockwise
                                    and points on the boundary are not removed
    T_res := ∅; t_last := connect(v_{n−1}, v₀);
    for i ∈ [0, n − 1] do begin
        let ⟨x₁, y₁⟩ = v_i, ⟨x₂, y₂⟩ = v_{(i+1) mod n}, t = connect(v_i, v_{(i+1) mod n})
        if (|x₁| < m ∧ |y₁| < m) ∨ (|x₂| < m ∧ |y₂| < m) ∧ θ(t) ≠ θ(t_last) then begin
            if (θ(t) − θ(t_last)) mod 2π = π ∧ |x₁| < m ∧ |y₁| < m then
                if y₁ = y₂ then T_res := T_res ∪ {sgn(x₁ − x₂)x ≤ sgn(x₁ − x₂)x₁}
                           else T_res := T_res ∪ {sgn(y₁ − y₂)y ≤ sgn(y₁ − y₂)y₁}
            T_res := T_res ∪ {t}; t_last := t;
        end
    end
    return T_res
end

function connect(⟨x₁, y₁⟩, ⟨x₂, y₂⟩)
    return (y₂ − y₁)x + (x₁ − x₂)y ≤ (y₂ − y₁)x₁ + (x₁ − x₂)y₁
```

Fig. 3. Convex hull algorithm for planar polyhedra

in $V_1 \cup V_2$. For each ray $r \in R$ we translate each vertex in $V_1 \cup V_2$ in the direction of the ray r. Note that the normalization of the rays and the translation by $2\sqrt{2}m$ ensures that the translated vertices are outside the square. We now apply the Graham convex hull algorithm [11], modified so that it removes all (strictly) interior vertices but retains points which lie on the boundary of the hull. What follows is a round-trip around this hull, translating two adjacent vertices into an inequality by calling *connect* if the following conditions are met: the inequality must have a different slope than the previously generated inequality and at least one of the two vertices must lie within the box. The two innermost if-statements deal with the pathological case of when V contains only colinear points and additional inequalities are needed to restrict the two opposing inequalities so that an (unbounded) line is not inadvertently generated.

The running time of this algorithm is dominated by the call to the convex hull algorithm of Graham [11] which takes $O(n \log n)$ time where $n = |V||R|$.

However, $|R|$ is at most eight (and usually between zero and four). Since $O(|V|) = O(|T|)$ it follows that the overall running time is $O((|T_1| + |T_2|)\log(|T_1| + |T_2|))$.

5 Projection

Projection returns the most precise system which does not depend on a given variable. We provide a constructive definition of projection for (complete) systems. Proposition 6 states that this coincides with the spatial definition of projection. Furthermore we prove that this operation preserves completion.

Definition 12. The operator $\exists_x : Two_X \to Two_{X\setminus\{x\}}$ is defined $\exists_x(T) = \cup\{\pi_Y(T) \mid Y = \{y,z\} \subseteq X \setminus \{x\}\}$.

Proposition 6. $\exists_x([T']_\equiv) = [\exists_x(T')]_\equiv$ and $\exists_x(T') \in Two'_X$ for all $T' \in Two'_X$.

Proof. By Fourier-Motzkin $\exists_x([T']_\equiv) = [T]_\equiv$ where $T = \{t \in T' \cup \mathit{result}(T') \mid x \notin \mathit{var}(t)\}$. Observe that $T \models \exists_x(T')$. Now suppose $r \in T' \cup \mathit{result}(T')$ such that $x \notin \mathit{var}(r)$. Then $T' \models r$, hence $\pi_{\mathit{var}(r)}(T') \models r$ and therefore $\exists_x(T') \models r$, and thus $\exists_x(T') \models T$, hence $\exists_x(T') \equiv T$ as required.

Now let $t \in Lin_X$ such that $\exists_x(T') \models t$. Moreover $T' \models \exists_x(T') \models t$, hence $\pi_{\mathit{var}(t)}(T') \models t$. Since $x \notin \mathit{var}(t)$, $\pi_{\mathit{var}(t)}(\exists_x(T')) \models t$ as required. ∎

Consider a complete system that includes $y - x \leq 0$ and $x - z \leq 0$. Projecting out x will preserve the inequality $y - z \leq 0$ which completion has made explicit.

6 Entailment

Entailment checking between systems of inequalities can be reduced to checking entailment on their two dimensional projections. Moreover, entailment checking for a planar polyhedron can be further reduced to checking entailment between three single inequalities. We start by detailing the entailment relationship between systems of inequalities and their two dimensional projections.

Proposition 7. Let $T' \in Two'_X$ and $T \in Two_X$. Then $T' \models T$ iff $\pi_Y(T') \models \pi_Y(T)$ for all $Y = \{x,y\} \subseteq X$.

Proof. Suppose $T' \models T$. Let $t \in \pi_Y(T)$. Then $T' \models T \models t$. Hence $\pi_{\mathit{var}(t)}(T') \models t$. Since $\mathit{var}(t) \subseteq Y$, $\pi_Y(T') \models t$ and therefore $\pi_Y(T') \models \pi_Y(T)$.

Now suppose $\pi_Y(T') \models \pi_Y(T)$ for all $Y = \{x,y\} \subseteq X$. Let $t \in T$. Then $t \in \pi_{\mathit{var}(t)}(T)$, hence $T' \models \pi_{\mathit{var}(t)}(T') \models \pi_{\mathit{var}(t)}(T) \models t$. ∎

Note that the proposition does not require both systems of inequalities to be complete. Due to Proposition 7 it suffices to check that entailment holds for all planar projections. Therefore consider checking entailment between two non-redundant planar systems $T_1, T_2 \in Two_{\{x,y\}}$. To test $T_1 \models T_2$ it is sufficient to show that $T_1 \models t$ for all $t \in T_2$. This reduces to finding $t_i, t_{i+1} \in T_1$ such that $\theta(t_i) \leq \theta(t) < \theta(t_{i+1})$ (modulo 2π). If any of the tests $\{t_i, t_{i+1}\} \models t$ fail, *false* can be returned immediately. If the inequalities are ordered by angle, planar entailment checking is linear time as shown in Fig. 4.

```
function entails(T_1 ∈ Two'_X, T_2 ∈ Two_X) begin
    if ∃t ∈ T_1 . t ≡ 0x + 0y ≤ −1 then return true;
    if ∃t ∈ T_2 . t ≡ 0x + 0y ≤ −1 then return false;
    let {t_1, ..., t_n} = T_1 such that θ(t_1) ≤ θ(t_2) ≤ ... ≤ θ(t_n);
    let {t'_1, ..., t'_m} = T_2 such that θ(t'_1) ≤ θ(t'_2) ≤ ... ≤ θ(t'_m);
    u := 1; l := n;
    for i ∈ [1, m] do begin
        while θ(t_u) < θ(t'_i) ∧ u ≤ n do begin
            l := u;
            u := u + 1;
        end
        if {t_l, t_(u mod n)} ⊭ t'_i then return false;
    end;
    return true;
end;
```

Fig. 4. Algorithm for checking entailment of planar polyhedra

7 Widening

For domains that do not satisfy the ascending chain property, widening is necessary to enforce termination of fixpoint calculations [7] (for example in loops). Widening can also be used to improve space and time behavior. In the following sections we elaborate on both.

7.1 Widening for termination

Any widening [7, 8] for polyhedra can be applied to planar polyhedra and then lifted to systems of two variables per inequality. Since the domain is structured in terms of projections, one tactic for delaying widening, and thereby improving precision, is to only apply widening when the number of projections has stabilized and the dimension of each of the projections is also stable. One subtlety is that applying completion after widening can compromise termination by reintroducing inequalities that were removed during widening.

7.2 Widening for tractability

To assess the tractability of the domain, we implemented a naïve completion operation and measured the growth both in the number of projections and inequalities. Our test data is obtained by generating random planar polytopes over different pairs of variables. Each polytope was constructed by computing the convex hull of a random set of points distributed across a square in $\mathbb{R}^2$. We set up three different scenarios called varying, constant and sparse. In the varying scenario, we created polytopes which had between 3 and 13 inequalities each until we reached 147 inequalities in total. To make the results comparable, we

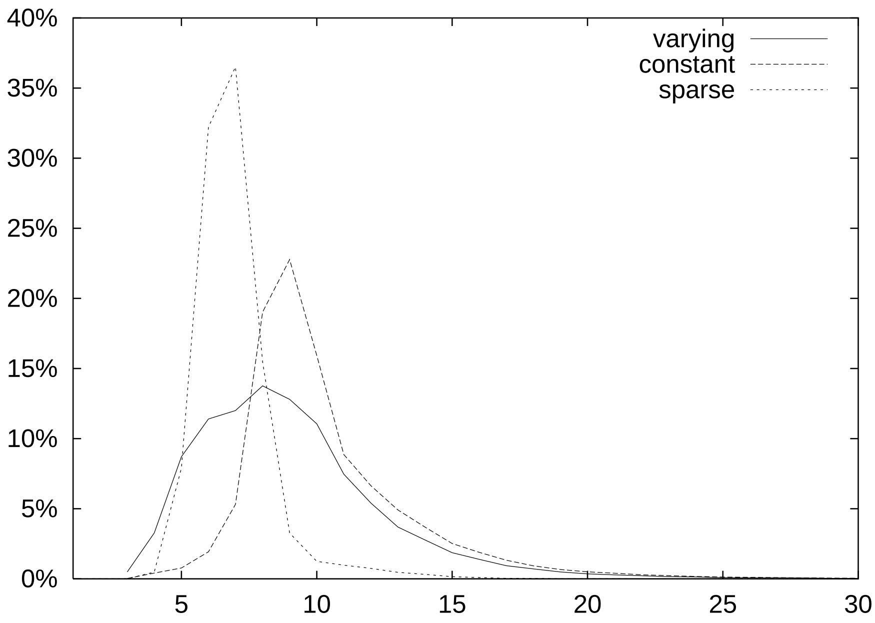


Fig. 5. The number of inequalities seems to be restricted in practice

then applied completion to those systems which had exactly 100 non-redundant inequalities. Redundancies can occur in the original system since two polytopes may share a common variable and a bound on this variable may propagate from one sub-system to the other, rendering inequalities superfluous. The constant scenario creates 10 inequalities for each pair of variables. Since fewer non-empty projections were initially generated (on average 143/10), the growth in the number of projections is larger – on average it increased to 32 projections. The last case, sparse, corresponds to a system where inequalities are weakly coupled, that is, few inequalities share variables. As expected the number of extra projections generated by completion is marginal. The results are summarized in Figure 6. Since randomly generated data offers no particular advantage to our completion algorithm over real data, it appears the completion will remain tractable in practice. In particular, the worst case quadratic growth in the number of projections is unlikely to arise.

An interesting observation is that the number of inequalities is not proportional to the number of points n over which the convex hull is calculated. This squares with probabilistic theory [5, 31]. Specifically, the convex hull of a set of n points randomly distributed over a square is expected to have $O(\log n)$ extreme points [5], while a random set of n points restricted to a circle is expected to have $O(n^{\frac{1}{3}})$ extreme points [31]. In our experiments, less than 1% of all projections had more than 30 inequalities (see Fig. 5 for the distribution). This suggests

scenario	*varying*	*constant*	*sparse*
dimension	10	10	100
inequalities generated	147	143	139
inequalities per polyhedron	3–13	10	10
after redundancy removal			
remaining inequalities	100	100	100
avg. no of ineq. per polyhedron	5.3	7.0	7.1
after completion			
avg. resultant inequalities	210	189	106
increase in no of projections	56%	123%	9%
projections > 30 inequalities	0.22%	0.18%	0.00%

Fig. 6. The impact of calculating completion

that pruning the number of inequalities down to a constant bound will have little overall effect on precision, yet obtains an attractive $O(d^3(\log d)^2)$ performance guarantee. One way to systematically drop inequalities is to remove those that contribute least to the shape, that is, remove the inequality that contributes the shortest edge to the polyhedron.

8 Future Work

Using union-find an arbitrary $T \in Two_X$ can be partitioned in near-linear time into a system $\{T_1, \ldots, T_p\}$ such that $var(T_i) \cap var(T_j) = \emptyset$ whenever $i \neq j$. This decomposition enables the complexity of completion to be reduced to $O(d^3(\log d)^2)$ where $d = \max\{|var(T_1)|, \ldots, |var(T_p)|\}$. This tactic, which is applicable to any polyhedral domain, will be useful if the coupling between variables is low.

The completion of a system T is currently computed iteratively in approximately $\log_2(|var(T)|)$ steps. The completion operation could benefit from applying a strategy such as semi-naïve iteration [3] that would factor out some of the repeated work.

9 Related work

The Octagon domain [26] represents inequalities of the form $ax_i + bx_j \leq c$ where $a, b \in \{1, 0, -1\}$ and $x_i, x_j \in X$. The main novelty of [26] is to simultaneously work with a set of positive variables x_i^+ and negative variables x_i^- and consider a DBM over $\{x_1^+, x_1^-, \ldots, x_d^+, x_d^-\}$ where $d = |X|$. Then $x_i - x_j \leq c$, $x_i + x_j \leq c$ and $x_i \leq c$ can be encoded respectively as $x_i^+ - x_j^+ \leq c$, $x_i^+ - x_j^- \leq c$ and $x_i^+ - x_i^- \leq 2c$. Thus an $2d \times 2d$ square DBM matrix is sufficient for this domain. Note that this DBM representation contains entries of the form $x_i^+ - x_j^+ \leq \infty$ whenever $x_i - x_j$ is not constrained (and likewise for $x_i + x_j \leq c$ and $x_i \leq c$). Closure is computed with an all-pairs Floyd-Warshall shortest-path algorithm that is $O(d^3)$ and echos ideas in the early work of Pratt [30]. Other earlier work on this theme considered the domain of inequalities of the form $x_i - x_j \leq c$ [25, 33], though the connection between bounded differences [9] and abstract interpretation dates back (at least) to Bagnara [1]. Very recently, Miné [27] has generalized DBMs to a class of domains that represent invariants of the form $x - y \in C$ where C is a non-relational domain that represents, for example, a congruence class [12]. This work is also formulated in terms of shortest-path closure and illustrates the widespread applicability of the closure concept.

Another thread of work is that of Su and Wagner [35] who propose a polynomial algorithm for calculating integer ranges as solutions to two variable per inequality systems, despite the intractability of some of these problems [21]. However, efficient integer hull algorithms do exist for the planar case [10, 14]. Combined with our completion technique, this suggests a new tractable way of calculating the integer convex hulls for two variable systems that promises to be useful in program analysis.

It is well-known that the linear programming problem – the problem of maximizing a linear function subject to linear inequalities – is polynomial time (Turing) equivalent to the problem of deciding whether a linear system is satisfiable. Moreover, the problem of deciding whether a linear system is satisfiable can be transformed into an equivalent problem where each inequality contains at most three variables (with at most a polynomial increase in the number of variables and inequalities). Thus an efficient algorithm for solving this problem is also an efficient algorithm for solving the linear programming problem and vice versa. This equivalence, and negative results such as [20], explains the interest in checking the satisfiability of systems of linear inequalities where each inequality contains at most two variables that dates back to [29, 30, 34]. Of all the proposals for checking the satisfiability of a system T, the algorithm of [16] is most in tune with the requirements of abstract interpretation due to its succinctness and its $O(|T||var(T)|^2 \log(|T|))$ running time which is guaranteed without widening. This result (and related results) provide fast entailment checking algorithms which may be useful for efficient fixpoint detection.

The trade-off between expressiveness and tractability is also an important consideration in constraint solving and in this context the class of two variables per inequality has also received attention [15, 18]. Jaffar *et al* [18] extend the closure algorithm of Shostak [34] for checking satisfiability over the reals to the integers by alternating closure with a tightening operation. However, this procedure is not guaranteed to either terminate nor detect satisfiability. Jaffar *et al* [18] go onto show that two-variables per inequality constraints with unit coefficients can be solved in polynomial time and that this domain supports efficient entailment checking and projection. More recently, Harvey and Stuckey [15] have shown how to reformulate this solver to formally argue completeness.

10 Conclusion

We proposed a new abstract domain of linear inequalities where each of the inequalities has at most two variables and the coefficients are unrestricted. We have shown how a (polynomial) completion operation leads to efficient and simple domain operations. Empirical evidence was presented that suggests that the domain is both tractable and well suited to widening.

Acknowledgments

We thank Roberto Bagnara, Les Hatton, Peter Linnington, Mike Norrish, Antoine Miné, Justin Pearson and Warwick Harvey for interesting discussions on polyhedra libraries and linear inequalities. We also thank the anonymous referees for their comments.

References

1. R. Bagnara. *Data-Flow Analysis for Constrant Logic-Based Languages.* PhD thesis, Dipartimento di Informatica, Università di Pisa, 1997.
2. V. Balasundaram and K. Kennedy. A Technique for Summarizing Data Access and its Use in Parallelism Enhancing Transformations. In *Programming Language Design and Implementation*, pages 41–53. ACM Press, 1989.
3. F. Bancilhon and R. Ramakrishnan. An Amateur's Introduction to Recursive Query Processing Strategies. In *International Conference on Management of Data*, pages 16–52. ACM Press, 1986.
4. F. Benoy and A. King. Inferring Argument Size Relationships with CLP($\mathbb{R}$). In *Logic Program Synthesis and Transformation (Selected Papers)*, volume 1207 of *Lecture Notes in Computer Science*, pages 204–223. Springer-Verlag, 1997.
5. J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the Average Number of Maxima in a Set of Vectors. *Journal of the ACM*, 25:536–543, 1978.
6. N. V. Chernikova. Algorithm for Discovering the Set of All Solutions of a Linear Programming Problem. *USSR Computational Mathematics and Mathematical Physics*, 8(6):282–293, 1968.
7. P. Cousot and R. Cousot. Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation. In *Programming Language Implementation and Logic Programming*, volume 631 of *Lecture Notes in Computer Science*, pages 269–295. Springer-Verlag, 1992.
8. P. Cousot and N. Halbwachs. Automatic Discovery of Linear Restraints among Variables of a Program. In *Principles of Programming Languages*, pages 84–97. ACM Press, 1978.
9. E. Davis. Constraint Propagation with Interval Labels. *Artificial Intelligence*, 32(3):281–331, 1987.
10. S. D. Feit. A Fast Algorithm for the Two-Variable Integer Programming Problem. *Journal of the ACM*, 31(1):99–113, 1984.
11. R. L. Graham. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters*, 1(4):132–133, 1972.

12. P. Granger. Static Analysis of Linear Congruence Equalities among Variables of a Program. In *International Joint Conference on the Theory and Practice of Software Development*, volume 493 of *Lecture Notes in Computer Science*, pages 169–192. Springer-Verlag, 1991.
13. W. H. Harrison. Compiler Analysis of the Value Ranges for Variables. *IEEE Transactions on Software Engineering*, SE-3(3), 1977.
14. W. Harvey. Computing Two-Dimensional Integer Hulls. *SIAM Journal on Computing*, 28(6):2285–2299, 1999.
15. W. Harvey and P. J. Stuckey. A Unit Two Variable per Inequality Integer Constraint Solver for Constraint Logic Programming. *Australian Computer Science Communications*, 19(1):102–111, 1997.
16. D. S. Hochbaum and J. Naor. Simple and Fast Algorithms for Linear and Integer Programs with Two Variables per Inequality. *SIAM Journal on Computing*, 23(6):1179–1192, 1994.
17. J. M. Howe and A. King. Specialising Finite Domain Programs using Polyhedra. In *Logic Programming, Synthesis and Transformation (Selected Papers)*, volume 1817 of *Lecture Notes in Computer Science*, pages 118–135. Springer-Verlag, 1999.
18. J. Jaffar, M. J. Maher, P. J. Stuckey, and R. H. C. Yap. Beyond Finite Domains. In *International Workshop on Principles and Practice of Constraint Programming*, volume 874 of *Lecture Notes in Computer Science*, pages 86–94. Springer-Verlag, 1994.
19. M. Karr. Affine Relationships Among Variables of a Program. *Acta Informatica*, 6:133–151, 1976.
20. V. Klee and G. J. Minty. How Good is the Simplex Algorithm? In *Inequalities – III.* Academic Press, New York and London, 1972.
21. J. C. Lagarias. The Computational Complexity of Simultaneous Diophantine Approximation Problems. *SIAM Journal on Computing*, 14(1):196–209, 1985.
22. K. G. Larsen, J. Pearson, C. Weise, and W. Yi. Clock Difference Diagrams. *Nordic Journal of Computing*, 6(3):271–298, 1999.
23. H. Le Verge. A Note on Chernikova's Algorithm. Technical Report 1662, Institut de Recherche en Informatique, Campus Universitaire de Beaulieu, France, 1992.
24. N. Lindenstrauss and Y. Sagiv. Automatic Termination Analysis of Logic Programs. In *International Conference on Logic Programming*, pages 63–77. MIT Press, 1997.
25. A. Miné. A New Numerical Abstract Domain Based on Difference-Bound Matrices. In *Programs as Data Objects*, volume 2053 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2001.
26. A. Miné. The Octagon Abstract Domain. In *Eighth Working Conference on Reverse Engineering*, pages 310–319. IEEE Computer Society, 2001.
27. A. Miné. A Few Graph-Based Relational Numerical Abstract Domains. In *Ninth International Static Analysis Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 117–132. Springer-Verlag, 2002.
28. J. Møller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard. Difference Decision Diagrams. In *Conference of the European Association for Computer Science Logic*, volume 1683 of *Lecture Notes in Computer Science*, pages 111–125. Springer-Verlag, 1999.
29. C. G. Nelson. An $n^{\log(n)}$ Algorithm for the Two-Variable-Per-Constraint Linear Programming Satisfiability Problem. Technical Report STAN-CS-78-689, Stanford University, Department of Computer Science, 1978.
30. V. R. Pratt. Two Easy Theories Whose Combination is Hard, September 1977. http://boole.stanford.edu/pub/sefnp.pdf.

31. H. Raynaud. Sur L'enveloppe Convexe des Nuages de Points Aléatoires dans $\mathbb{R}^n$. *Journal of Applied Probability*, 7(1):35–48, 1970.
32. R. Seidel. Convex Hull Computations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 361–376. CRC Press, 1997.
33. R. Shaham, H. Kolodner, and M. Sagiv. Automatic Removal of Array Memory Leaks in Java. In *Compiler Construction*, volume 1781 of *Lecture Notes in Computer Science*, pages 50–66. Springer, 2000.
34. R. Shostak. Deciding Linear Inequalities by Computing Loop Residues. *Journal of the ACM*, 28(4):769–779, 1981.
35. Z. Su and D. Wagner. Efficient Algorithms for General Classes of Integer Range Constraints, July 2001. http://www.cs.berkeley.edu/~zhendong/.
36. D. Wagner, J. S. Foster, E. A. Brewer, and A. Aiken. A First Step Towards Detection of Buffer Overrun Vulnerabilities. In *Network and Distributed System Security Symposium*. Internet Society, 2000.