



City Research Online

City St George's, University of London

Citation: Brain, M. & Howe, J. M. (2026). Canonical Forms and Widening for Two Variables Per Inequality Systems. *Mathematics in Computer Science*, 20(1), 2. doi: 10.1007/s11786-025-00612-6

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: <https://openaccess.city.ac.uk/id/eprint/34800/>

Link to published version: <https://doi.org/10.1007/s11786-025-00612-6>

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to [City Research Online policy](#).

Canonical Forms and Widening for Two Variables Per Inequality Systems

Martin Brain^{1*} and Jacob M. Howe^{1*}

¹Department of Computer Science, City St George's, University of
London, London, EC1V 0HB, United Kingdom.

*Corresponding author(s). E-mail(s): martin.brain@city.ac.uk;
j.m.howe@city.ac.uk;

Abstract

Program verification using abstract interpretation involves the symbolic calculation of fixpoints over lattices. Integral to these fixpoint calculations is widening, an operation that trades precision for guarantees of termination. Abstract interpretation often works with lattices of convex sets of points in n-dimensional space, represented by sets of linear inequalities. When the form of these inequalities is restricted these are known as weakly relational domains. This paper addresses weakly relational domains, the detail of their representation and the way in which widening is applied, including study of how the closure operators used in weakly relational domains interact with widening, and considers how sequences of constraints might be widened. Satisfiability checking for numeric constraints is one tool used in this work.

Keywords: Abstract interpretation, widening, weakly relational domains

1 Introduction

This paper concerns the widening operator used in abstract interpretation based static program analysis. In particular, it investigates how widening interacts with the representation of weakly relational domains.

Abstract interpretation [1, 2] is an approach to formal program analysis. A program is abstracted, with respect to a property of interest, into a mathematical representation. The underlying structure is called an abstract domain taking the form of a

partially ordered set, typically a lattice. Values at each program point can be represented as a point in the lattice. The abstract program is executed, updating the representation at each program point until a fixpoint is reached. This fixpoint will be an over-approximation of the least fixpoint of the program. Interpreting the abstract fixpoint back into the language of the program allows analysis of the original program.

Abstract domains based on polyhedra have been considered since the introduction of abstract interpretation [3]. However, the domain operations for polyhedra are prohibitively expensive for practical analysis tools. This has led to interest in subclasses of polyhedra called *weakly relational domains*. These restrict the constraints in the abstract domain in some way, typically allowing at most *two variables per inequality* (TVPI, [4]) and often also restricting the coefficients of the variables in the inequalities, for example to +1,0,-1 to give *Octagons* [5]. The aim of weakly relational domains is to have more tractable domain operations, whilst retaining sufficient precision that analysis does not overwhelm the user with spurious warnings.

Polyhedra (and their subclasses) are convex sets of points in n-dimensional space. Polyhedra can be considered in two ways. The first is spatially, as a set of points (clearly, this is an infinite representation). The second is syntactically, as a representation of this space. Various finite representations of polyhedra exist. The spatial consideration provides the semantics and is in some sense the primary way to think about polyhedra in the context of abstract interpretation. However, the syntactic view, the representation, is the object of study when systems are being built. The differences between these two will be of importance here. The syntactic representation of concern is as sets of linear inequalities (with points being solutions to this set of constraints). There will be some focus on unary constraints, leading to a canonical form for TVPI constraints.

As well as linear constraints, abstract domains can be based on many things, for example, sets, Boolean functions or non-linear constraints. In many cases, and in particular for linear constraints, it is possible that the lattice of the abstract domain contains infinite ascending chains, hence termination of fixpoint calculations is not guaranteed. Even if termination is guaranteed, long chains might mean that termination does not occur in practice. Abstract interpretation comes equipped with the concept of widening, where information is discarded during the calculation of a fixpoint. This “jumps” up the lattice and accelerates the termination of the abstract execution, potentially at the cost of precision of the resulting fixpoint.

Weakly relational domains usually maintain their representations in a closed form [5, 6], where implied inequalities are made explicit, allowing join, entailment and satisfiability operations to be reduced to planar calculations. This approach is potentially problematic when considered with widening. Widening typically throws away inequalities that are in some way unstable across iterations, moving the abstraction up the lattice with the hope of quickly reaching a fixpoint. However, it is possible that an inequality discarded was one introduced when calculating the closed form. This would then be immediately re-introduced, preventing the computation reaching a fixpoint.

This paper is about widening, with a particular focus on weakly relational domains, and extends [7]. The representation of TVPI constraints is considered, giving a canonical form for TVPI systems. A characterisation of a variation of the classical approach

to widening in the context of weakly relational domains is given. Then attention is paid to what widening is, noting the importance of the linear widening problem: how to find over-approximations for sequences of constraints with the same coefficients. A series of problems are analysed with attention paid to the issues discussed.

This paper makes the following contributions:

- Provides a canonical form for systems of TVPI constraints
- Characterises an approach to widening for weakly relational domains
- Uses one dimensional sequence prediction to give more precise widening
- Investigates a series of examples of widening, illustrating problems and some solutions to the application of widening.

The paper is structured as follows: Section 2 gives background and related work on weakly relational domains and widening, including a motivating worked example. Section 3 provides further background on weakly relational domains and gives a canonical form for TVPI systems. Section 4 gives a new characterisation of widening for closed systems of TVPI constraints. Section 5 uses sequence prediction to provide a more precise widening. Section 6 considers examples in the light of the above. Section 7 concludes.

2 Background and Related Work

This work is concerned with widening over abstract domains consisting of sets of linear inequalities, or polyhedral domains. This section gives background on polyhedra and their subclasses, a brief review of widening, details on weakly relational domains and closure, before giving an example of code analysis using weakly relational domains.

2.1 Polyhedra and Weakly Relational Domains

The abstract domains of interest in this paper are lattices over sets of inequalities. Several domains are considered, varying in the form of inequalities that are allowed. The first of these is *Polyhedra*, Poly_X , the set of all linear inequalities over a set of variables X . Here (and through most of this paper), coefficients and constants are rationals, but other choices, including reals, are available.

Definition 1. $\text{Poly}_X = \{a_1x_1 + \dots + a_nx_n \leq e \mid x_1, \dots, x_n \in X \wedge a_1, \dots, a_n, e \in \mathbb{Q}\}$

Then $\langle \mathcal{P}(\text{Poly}_X), \sqsubseteq, \sqcup, \sqcap \rangle$ is a complete lattice, that is, sets of polyhedral constraints (implicitly quotiented by equivalence), or rather the solutions to those constraints, are ordered by inclusion (\sqsubseteq), with greatest lower bound (meet) being geometric intersection (\sqcap) and least upper bound (join) being convex hull (\sqcup). Observe that this lattice has infinite ascending chains. The sets of inequalities below lift to lattices in the same way. As well as the lattice operations above, the projection of a set of constraints onto constraints over a subset of X is an important operation in abstract interpretation.

Abstract interpretation using the Poly_X domain is seen as impractical beyond small programs as the size of the representation and the cost of performing operations

such as convex hull might become prohibitively large. Depending on the structure of the program, analysis using Poly_X may be undecidable [8, 9]. Therefore abstract domains based on restricted forms of linear inequalities have been investigated; that is, the relationships between variables that can be expressed have been restricted. Hence, these are referred to as weakly relational domains. Many weakly relational domains have been investigated, and those covered below are not intended to be a complete list. These (typically) restrict inequalities so that they have at most two variables in them. This then means that operations such as least upper bound can be calculated by a series of planar operations; since these planar operations have good computational complexity the intractability of full polyhedra is avoided. The most general version is then the two variables per inequality (TVPI) abstract domain [4, 6].

Definition 2. $\text{TVPI}_X = \{ax + by \leq e \mid x, y \in X \wedge a, b, e \in \mathbb{Q}\}$

The domain of *Logahedra* [10] restricts coefficients to be powers of 2 (potentially with a maximum power). Structures based on Logahedra have been used in the verification of embedded real-time systems [11].

Definition 3. $\text{Log}_X = \{ax + by \leq e \mid x, y \in X \wedge a, b \in \{-2^n, 0, 2^n \mid n \in \mathbb{Z}\} \wedge e \in \mathbb{Q}\}$

Octagons [5], where the inequalities have unit or zero coefficients, have received considerable attention and have been widely used for abstract interpretation based formal verification. For example, the Octagon abstract domain is an important component of the ASTRÉE static analyser, developed to verify the absence of classes of run-time errors in embedded C code [12]. This analyser has been used for avionics code by Airbus.

Definition 4. $\text{Oct}_X = \{ax + by \leq e \mid x, y \in X \wedge a, b \in \{-1, 0, 1\} \wedge e \in \mathbb{Q}\}$

Bounded differences (sometimes also called Zones) have also been popular [13, 14], partly because the constraint systems can naturally be represented as weighted graphs, leading to efficient algorithms (they are referred to as DBM for difference bounded matrices). Unary inequalities for DBM are typically dealt with by introducing a dummy variable for 0.

Definition 5. $\text{DBM}_X = \{x - y \leq e \mid x, y \in X \wedge e \in \mathbb{Q}\}$

Finally, using intervals as an abstract domain loses all ability to track relations, but does result in particularly scalable algorithms.

Definition 6. $\text{Int}_X = \{ax \leq e \mid x \in X \wedge a \in \{-1, 1\} \wedge e \in \mathbb{Q}\}$

When considering intervals, $x \in [a, b]$ is shorthand for $\{-x \leq -a, x \leq b\}$, whilst $x \in [a, \infty]$ is shorthand for $\{-x \leq -a\}$.

This paper primarily considers TVPI_X inequalities. Following [6], the intention is that algorithms developed for this domain will be inherited by abstract domains using subclasses of two variable inequalities, in particular Octagons.

2.2 Widening

Widening [1, 15] is an operation that forces the iterations of an abstract interpretation to reach a fixpoint, that is, for the analysis to terminate. This is formalised in the following definition.

Definition 7. Where L is a lattice, a *widening* is a function $\nabla : L \times L \rightarrow L$ such that:

1. $\forall x, y \in L. x \sqsubseteq x \nabla y$
2. $\forall x, y \in L. y \sqsubseteq x \nabla y$
3. for all increasing chains $x_0 \sqsubseteq x_1 \sqsubseteq \dots$, the chain given by $y_0 = x_0, y_{i+1} = y_i \nabla x_{i+1}$ is not strictly increasing.

Widening is about sequences of iterates, that is, it is about how the space of possible values of variables changes at a given program point as control returns to it (for example, in a loop). It is usual to merge the original abstraction at a widening point with the new value before applying widening (the definition hints that this is sensible), so $x \nabla y$ is really $x \nabla (x \sqcup y)$. Notice that widening is about more than controlled loss of precision within the representation, it is about enforcing termination. A particularly crude widening would then be to return the top element of the lattice. One point of interest is that by considering the topological structure of the abstract program (in particular strongly closed components), widening need only be applied, and termination checked, at certain nodes in the call graph [16].

For polyhedral analysis, widening as first introduced [3] allows the fixpoint calculation to iterate through a point twice, then on the third pass inequalities that are stable are retained and others are discarded. This allows a fixpoint to be reached. The paper uses the double description representation (where two representations of the space are maintained, one as vertices, rays and lines, the other as linear inequalities [17]; this contains an implicit assumption of non-redundancy), but variations on this, whatever the representation, remain the *classic approach* (terminology used throughout this paper). A number of heuristics as to how and when to widen are available and have been combined and refined in [18]. These include refraining from widening if it can be determined that a chain is finite, and more sophisticated delays of the application of widening.

The danger with the classic approach is that it widens too aggressively, jumping past more subtle fixpoints leading to an over-approximation that cannot verify a correctness condition. Various alternatives have been investigated. This includes widening with thresholds [19], where the user supplies, before analysis, a series of thresholds to widen to. So, in the classic approach $[1, 4] \nabla [1, 5] = [1, \infty]$, but if the user has supplied a threshold n (assuming $n \geq 5$), then $[1, 4] \nabla [1, 5] = [1, n]$. If the interval continues to expand, then once all thresholds are passed the classic widening applies and analysis will terminate; with some wisdom in the up front choice of threshold it might be

that the $[1, n]$ interval is a fixpoint, improving the calculated fixpoint and associated analysis.

This relies, to some extent, on the expertise and insight of the person conducting the analysis. In addition, the technique is typically limited to intervals (bounds on single variables), not capturing relational constraints. Another approach is to automate the selection of thresholds, as suggested in the widening with landmarks approach of [20, 21]. The lookahead widening of [22] is another attempt to find good values to widen to. Work on widening continues, with [23] considering how it can be used when analysing programs containing non-linear transformations, [24] tackling loops and how their analysis can be used to aid the successful application of widening, whilst [25] uses a binary search over solutions to a satisfiability problem to compute a minimal fixpoint. In [26] various specifications of widening are considered and potential dangers of mismatches in these specification identified.

2.3 Closure

When using weakly relational domains based on classes of inequalities such as TVPI or Octagons for abstract interpretation, some consideration is required as to how to represent and maintain the sets of inequalities. The choice of representation impacts on the cost of the domain operations, with the choice made as a trade off between the complexity of the operations. One of the major attractions of two variable systems is that they are closed under variable elimination: if a common variable is eliminated from a pair of two variable linear inequalities, the result is again a two variable linear inequality. Weakly relational domains often use a closed form, where some redundant inequalities are made explicit. After finding a closed system all non-redundant inequalities relating any pair of variables are made explicit. For example, suppose that $\{x - y \leq 1, 2y - 3z \leq 2\}$ is a system of inequalities, then the closed system also includes the redundant inequality $2x - 3z \leq 4$. The relationship between variables x, z has been made explicit. Closure as a concept is primarily syntactic, but becomes semantic when considering restrictions to a subset of variables.

A closed system means that a variable can be projected out of a system in constant time by simply dropping all constraints involving that variable. It also means that least upper bound computation can be reduced to two dimensional convex hull problems, likewise satisfiability can be computed at a planar level (plus entailment and equality are straightforward linear operations with the closed form). These good operations are traded off against the cost of maintaining the closed form. A typical program analysis step is to consider how a line of (abstracted) code updates the representation. This is done incrementally, so there is interest in how to update a closed representation upon the addition of a single new inequality (repeated applications of this step can be used to find a closed system from an initially unclosed system). Incremental closure for TVPI has been closely investigated in [6] where it is proved that any inequality made explicit is the result of at most two computation steps, whilst [27] considers incremental closure for zones.

The following reiterates some key definitions from [6]. Firstly, the set of variables that occur in inequality c is denoted $\text{vars}(c)$ and is defined:

Definition 8.

$$\text{vars}(ax + by \leq e) = \begin{cases} \emptyset & \text{if } x = y \wedge a = -b \\ \emptyset & \text{else if } a = b = 0 \\ \{y\} & \text{else if } a = 0 \\ \{x\} & \text{else if } b = 0 \\ \{x, y\} & \text{otherwise} \end{cases}$$

This allows the definition of syntactic projection of a system of inequalities onto a given subset of variables: simply select those inequalities including those variables.

Definition 9. The *syntactic projection*, denoted π_Y for some $Y \subseteq X$, of system of inequalities $S \subseteq \text{TVPI}_X$ is defined as $\pi_Y(S) = \{c \in S \mid \text{vars}(c) \subseteq Y\}$.

This is used in a formal definition of a closed system of inequalities. The entailment operation, \models , is geometric inclusion. If $C_1 \models C_2$ and $C_2 \models C_1$, then $C_1 \equiv C_2$. Notice that a closed system may include inequalities that are redundant even in a given projection (for example, unary constraints). Closure is a property of two variable systems. A system of inequalities over variables X is said to be closed if its restriction (its syntactic projection) onto any subset of X is equivalent to its spatial projection onto these variables, that syntactic and semantic projection coincide:

Definition 10. A system $C \subseteq \text{TVPI}_X$ is *closed* if the following predicate holds:

$$\text{closed}(C) \iff \forall c \in \text{TVPI}_X. (C \models c \Rightarrow \pi_{\text{vars}(c)}(C) \models c)$$

The representation of closed systems and their interaction with widening is investigated in the coming sections.

2.4 Worked Example

This example works through the analysis of a small section of code to illustrate the use of abstract interpretation using weakly relational domains for program verification. For a full introduction to abstract interpretation see [2].

Consider the following implementation of a C standard library function, with line numbers:

```
(1) char *strdup(const char *s) {
(2)     size_t n = strlen(s);
(3)     char *result = malloc(n+1);
(4)
(5)     size_t i = 0;
(6)     size_t j = 0;
(7)
(8)     while (i < n) {
(9)         result[j] = s[i];
(10)        i = i + 1;
```

```

(11)   j = j + 1;
(12)   }
(13)   result[j] = '\0';
(14)
(15)   return result;
(16)   }

```

If it is known that `s` points to a valid C string and `strlen` returns its length then this is clearly memory-safe. When reading from `s`, $i \in [0, n - 1]$ and when writing to `result` $j \in [0, n]$. Showing this via abstract interpretation is not as simple as it might appear.

First consider using the $\text{Int}_{\{i,j,n\}}$ abstract domain, and assume no hidden dependencies. At the start of the loop on Line (8) observe that $i \in [0, 0], j \in [0, 0], n \in [0, \infty]$, as in Figure 1 a).

Each instruction in the code has an abstract equivalent transforming the lattice point as passed to that line. For example, on Line (10) the increment on the program variable `i` will correspondingly shift the polyhedron representing the program state at this point by one in the `i` direction. At the end of the first iteration around the loop, at Line (12) $i \in [1, 1], j \in [1, 1]$. Control returns to the start of Line (8) and the new values for i, j need to be merged with the previous abstraction at this program point. The start of the loop on Line (8) can be reached either as control passes from Line (6), or as control returns to the head of the loop from Line (12). The abstract domain value at this point needs to merge these two control paths, summarising the values that can be taken. This is achieved by computing the least upper bound of the existing abstract domain value with the values coming from the loop. The merge results in new intervals for the variables, $i \in [0, 1], j \in [0, 1]$. This represents a square section as illustrated in Figure 1 b). Note that this region includes points like $(0, 1, 0)$ which are not possible and are only present because the over-approximation is unable to express the relation between variables.

Repeating this will result in the intervals for the variables at Line (8) increasing. However, this process will not terminate, since the values will not stabilise at a fixpoint. This necessitates widening. To accelerate (or indeed to reach) a fixpoint, information is discarded or generalised, until the abstract domain value at the merge point is stable. As discussed in Section 2.2, the classic way to apply widening is to execute the abstract loop two or three times, then to observe which constraints are stable from one iteration to the next, retaining only these. The result is a fixpoint. Note that this is often a syntactic observation, and that two semantically equivalent, but syntactically different, points might well result in widening being applied.

Using interval domain $\text{Int}_{\{i,j,n\}}$, for i and j only the lower bounds are stable across iterations, hence widening applies:

$$\begin{aligned}
& \{i \in [0, 1], j \in [0, 1], n \in [0, \infty]\} \nabla \{i \in [0, 2], j \in [0, 2], n \in [0, \infty]\} \\
& = \{i \in [0, \infty], j \in [0, \infty], n \in [0, \infty]\}
\end{aligned}$$

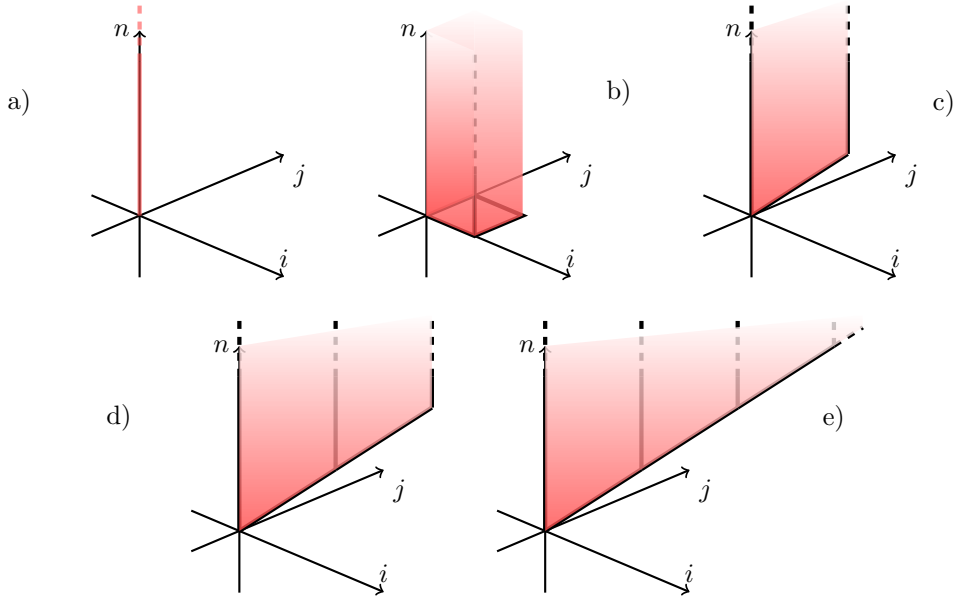


Fig. 1 Fixpoint iterations over $\text{TVPI}_{\{i,j,n\}}$, with widening

That is, the entire positive octant. This is a fixpoint (indeed the least fixpoint for this choice of abstract domain), but this does not verify memory safety at Line (13). An analysis using this would throw a spurious warning to the developer.

The Int abstract domain does not track relations between variables. The verification failure above suggests a need to track (some) relations between variables. Equalities might make a tempting choice of abstract domain: $i = j$ is a loop invariant at the head of the loop, however it is not maintained during the loop; at the start of Line (11) $i = j + 1$, so something more expressive than equalities is needed.

Consider using the $\text{TVPI}_{\{i,j,n\}}$ domain. At the first merge point considered above, the same inequalities are merged. The least upper bound calculation together with closure results in $\{-i \leq 0, -j \leq 0, -n \leq 0, i - n \leq 0, i - j \leq 0, j - i \leq 0, j - n \leq 0, i \leq 1, j \leq 1\}$. This represents a section of plane reaching through $i = j$ (see Figure 1 c)). Repeating this will result in the plane section increasing in size (as in Figure 1 d)). At the next iteration widening is applied, retaining only the stable inequalities $\{-i \leq 0, -j \leq 0, -n \leq 0, i \leq n, i - j \leq 0, j - i \leq 0, j - n \leq 0\}$ as illustrated in Figure 1 e). This fixpoint implies that $j \leq n$ (indeed in the closed representation used this is explicit) hence the array access on Line (13) can be verified. This would be the case if unrestricted polyhedra were used, but also if a less expressive weakly relational domain such as $\text{Oct}_{\{i,j,n\}}$, or even bounded differences, $\text{DBM}_{\{i,j,n\}}$, were used since all the inequalities in this example are in these classes of constraints.

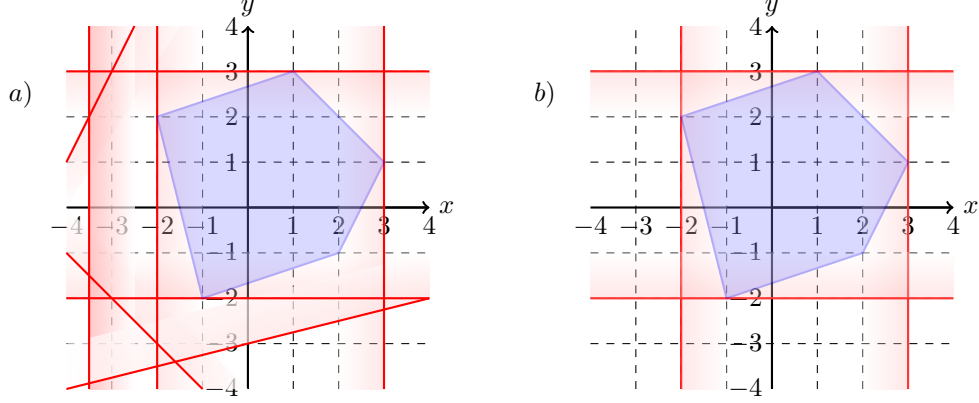


Fig. 2 Examples of closed systems

3 Canonical Form

This section provides a canonical form for closed TVPI systems, that is, a unique representation for each point in the TVPI lattice. It first considers the syntactic representation issue of closed systems, notes that previous work addressing this issue has failed to provide a canonical form, then provides such a canonical form. In later sections the interaction between closed systems, the canonical form defined, and widening will be investigated.

3.1 Uniqueness

Consider Figure 2 which gives two examples of closed systems over just two variables. These represent the following three sets of inequalities:

$$X = \{x + y \leq 4, x - 3y \leq -8, -4x - y \leq 6, -x + 3y \leq -5, -2x + y \leq -5\}$$

$$B = \{x \leq 3, y \leq 3, -x \leq 2, -y \leq 2\}$$

$$R = \{2x - y \leq -9, -x \leq \frac{7}{2}, -x - y \leq 7, -x + 4y \leq -12\}$$

In Figure 2 a) $X \cup B \cup R$ is represented. As can be seen, the inequalities in X define the blue space, and removal of any of these constraints changes the space defined. The inequalities in B define the bounds for the values of x and y . They are redundant in the 2d space pictured, in sense that their removal would not change the space defined, however the syntactic projections onto either x or y requires these bounds to be retained in order for the system to be closed. The inequalities in R are redundant, and in addition they do not describe the unary bounds for the system. The system $X \cup B \cup R$ is closed according to the Definition 10.

It can be seen that the inequalities in R are not necessary for the system to be closed. Their removal does not change the space defined, nor does it change the syntactic projection of the system on to a single variable (as is the case for those in B). That is, $X \cup B$ is a smaller, also closed, system (see Figure 2 b)). Indeed, in this example, it is the minimal closed system in the sense that all the inequalities must be included in a closed system and removing any of these inequalities would result in a system that is not closed.

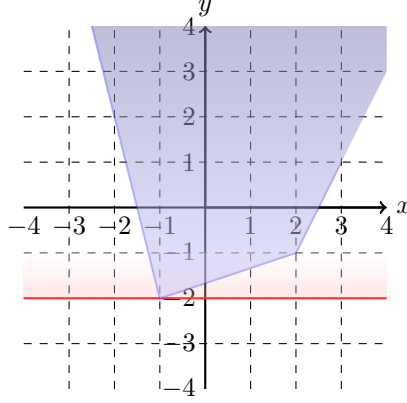


Fig. 3 Example of an unbounded closed system

Figure 3 illustrates another closed system, $\{-4x - y \leq 6, -x + 3y \leq -5, -2x + y \leq -5, -y \leq 2\}$, this time one which is not bounded in every direction.

Now consider Figure 4. This illustrates representations of a line segment that continues unbounded in the upper right quadrant. In Figure 4 a) $C = \{x - y \leq 0, -x + y \leq 0, -x - 2y \leq 0, -2x - y \leq 0\}$ is represented. First notice that this is not closed since (for example), $\{-x + y \leq 0, -x - 2y \leq 0\} \models -x \leq 0$, but $-x \leq 0 \notin C$. In Figure 4 b) $C = \{x - y \leq 0, -x + y \leq 0, -x \leq 0, -y \leq 0\}$ is represented. This is closed; notice that only one of the bounds is required to define the space, but the other is required for closure hold.

Similarly, a point might be represented in a infinite number of ways, as illustrated in Figure 5. The first representation (Figure 5 a)) is not closed, whilst the second (Figure 5 b)) is, but notice that the union of the two systems is also closed.

The examples above have illustrated two points. Firstly, that for a system to be closed the tightest unary bounds need to be present in the systems, and secondly that closed representations are not unique.

The issue of uniqueness of representation is particularly acute for TVPI. Work on other weakly relational domains such as Oct or DBM has represented domain elements using matrix or graph structures, rather than systems of inequalities. These are then maintained so that all of the inequalities in each projection are represented (noting that there are eight possibilities per projection for Oct, six for DBM). This makes bounds explicit, as well as maintaining potentially redundant two variables inequalities.

These matrix and graph representations are not available (or, at least, not obvious) for TVPI, since the number of templates for constraints is not finite. It would be desirable for the representation of two variable inequality systems to have a canonical form – every space that can be described has a unique representation.

3.2 Unary Inequalities and Canonical Form

In [6] it was recognised that redundant inequalities should be removed and that it is important to explicitly represent bounds (even if they are redundant). The following

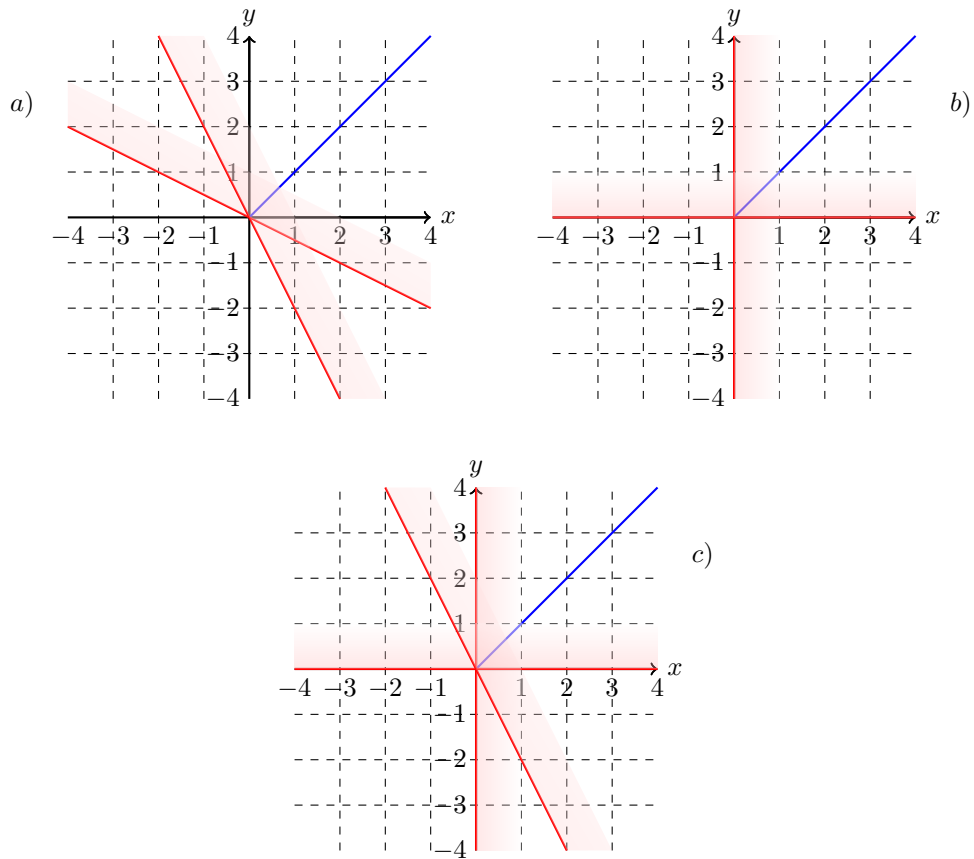


Fig. 4 Example of a line segment

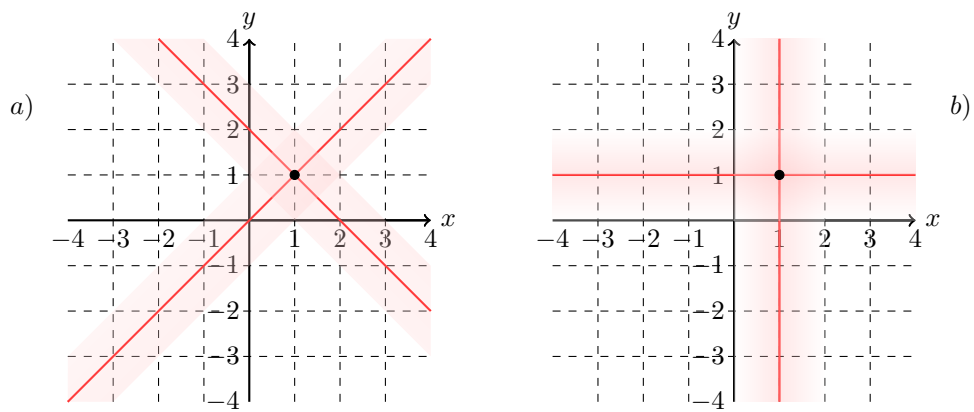


Fig. 5 Example of a point

operation (slightly adapted from [6]) applies to systems of TVPI constraints and removes redundant inequalities whilst maintaining bounds.

Definition 11. The mapping $\text{filter} : \mathcal{P}(\text{TVPI}_X) \rightarrow \mathcal{P}(\text{TVPI}_X)$ is defined as follows, where $C \in \text{TVPI}_X$

$$\text{filter}(C) = \cup_{Y \subseteq X, |Y| \leq 2} \text{filter}_Y(\pi_Y(C))$$

where for all $Y \subseteq X, |Y| \leq 2$

1. $\text{filter}_Y(C) \subseteq C$
2. $\text{filter}_Y(C) \equiv \pi_Y(C)$
3. for every $C' \subset \text{filter}_Y(C)$, $C' \not\equiv \text{filter}_Y(C)$.

However, filter is not quite strong enough to enforce a unique representation when the system is not fully dimensional. It treats the one variable and two variable projections separately ignoring the importance of the unary constraints in the two variable projection. Returning to the line segment example of Figure 4, if $C = \{x - y \leq 0, -x + y \leq 0, -x - 2y \leq 0, -2x - y \leq 0, -x \leq 0, -y \leq 0\}$ (the union of the two systems) then $\text{filter}_{\{x\}}(C) = \{-x \leq 0\}$ and $\text{filter}_{\{y\}}(C) = \{-y \leq 0\}$. But for the $\{x, y\}$ projection itself there is no requirement for $\text{filter}_{\{x, y\}}$ to retain bounds. Hence there are four possible values of $\text{filter}_{\{x, y\}}(C)$ one of which is $\text{filter}_{\{x, y\}}(C) = \{x - y \leq 0, -x + y \leq 0, -x - 2y \leq 0\}$. Then $\text{filter}(C) = \text{filter}_{\{x, y\}}(C) \cup \text{filter}_{\{x\}}(C) \cup \text{filter}_{\{y\}}(C)$ contains an extraneous $-x - 2y \leq 0$, as illustrated in Figure 4 c).

The following definition of canon enforces a stronger link between the one and two variable projections and leads to a unique representation for closed systems.

Definition 12. The mapping $\text{canon} : \mathcal{P}(\text{TVPI}_X) \rightarrow \mathcal{P}(\text{TVPI}_X)$ is defined as follows, where $C \in \text{TVPI}_X$ and $\text{closed}(C)$ holds.

$$\text{canon}(C) = \cup_{Y \subseteq X, |Y| \leq 2} \text{canon}_Y(\pi_Y(C))$$

where for all $Y \subseteq X, |Y| \leq 2$

1. $\text{canon}_Y(C) \subseteq C$
2. $\text{canon}_Y(C) \equiv \pi_Y(C)$
3. if $|Y| = 1$ then for every $C' \subset \text{canon}_Y(C)$, $C' \not\equiv \text{canon}_Y(C)$.
4. if $|Y| = 2$ and $Y = \{x, y\}$, then for every $C' \subset \text{canon}_Y(C) \setminus \text{canon}_{\{x\}}(C) \setminus \text{canon}_{\{y\}}(C)$, $C' \cup \text{canon}_{\{x\}}(C) \cup \text{canon}_{\{y\}}(C) \not\equiv \text{canon}_Y(C)$.

Case 1 says that the constraints are drawn from the closed system. Case 2 the output is equivalent to the input in every projection (including the single variable projections). Case 3 says that only the tightest bounds on single variables are represented. Case 4 says that the output is minimal in the sense that if a non-unary constraint is removed it will change the space represented.

The following proposition states that canon provides a canonical form for closed TVPI systems, that is, two semantically equivalent closed systems have the same representation syntactically. In the below, $\neg(ax + by \leq e) \equiv ax + by > e$, with this notation only used to reason geometrically about satisfiability; strict inequalities are

not part of the two variable systems under investigation.

Proposition 1. If $C_1 \equiv C_2$, $\text{closed}(C_1)$ and $\text{closed}(C_2)$, then $\text{canon}(C_1) = \text{canon}(C_2)$.

Proof. Case 3 of the definition of canon ensures that one variable projections have a unique representation. Since the systems considered are closed, it is enough to consider two variable projections for the result to lift to arbitrary dimension.

Observe that a single point has a unique representation, since the projections onto its two individual dimensions have unique representations as a pair of bounds and that any non-unary inequality is then redundant and omitted by Case 4 of the definition of canon . Similarly a line segment is uniquely represented as the pair of inequalities representing the line together with unary inequalities segmenting it as appropriate.

Suppose that there is a fully two dimensional space with $\text{canon}(C_1) \equiv \text{canon}(C_2)$, but $\text{canon}(C_1) \neq \text{canon}(C_2)$. Then (without loss of generality) there must be some $c \in \text{canon}(C_1)$ such that $c \notin \text{canon}(C_2)$. Hence $\text{canon}(C_2) \models c$ and $\text{canon}(C_1) \setminus \{c\} \not\models \text{canon}(C_1)$. The latter implies that $\text{canon}(C_1) \cup \{-c\}$ is satisfiable, whilst the former implies that $\text{canon}(C_1) \cup \{-c\}$ is unsatisfiable. This contradicts $\text{canon}(C_1) \equiv \text{canon}(C_2)$, therefore $\text{canon}(C_1) = \text{canon}(C_2)$. \square

Returning to the example of the line segment in the positive quadrant from Figure 4, where the closed system is the union of Figure 4 a) and b), observe that Cases 1 and 2 of the definition of canon means that $x \geq 0, y \geq 0, x - y \leq 0, y - x \leq 0$ must be in the canonical system, whilst Case 4 ensures that other redundant constraints are omitted, hence the canonical system is that illustrated in Figure 4 b).

4 Widening and Closure

There is a problem when using widening whilst calculating a fixpoint over weakly relational domains. Widening attempts to weaken a system of inequalities by discarding inequalities that are not displaying stability. This is syntactically calculated: consider A_i and A_{i+1} as sets of inequalities, then $A_i \nabla A_{i+1} = A_i \cap A_{i+1}$. To maintain a closed representation the result needs to be made closed, and in doing so weakly relational domains are introducing redundant inequalities into the representation. The interaction of these two is unclear, but an infinite loop of discarding an inequality that is immediately reintroduced by closure is possible, and is described in Section 4.2.

4.1 Incremental Completion

The process of generating implied inequalities is based on the result operator that eliminates a shared variable from a pair of inequalities. The process of finding a closed system, including an interleaved with applications of canon to remove unnecessary inequalities, is called *completion*. In the definition of complete below, result will be lifted to sets of inequalities, referring to all pairwise applications of result .

Definition 13. If $c_1 \equiv a_1x + b_1y \leq e_1$, $c_2 \equiv a_2x + b_2z \leq e_2$ and $a_1a_2 < 0$ then

$$c = \text{result}(c_1, c_2, x) = |a_2|b_1y + |a_1|b_2z \leq |a_2|e_1 + |a_1|e_2$$

otherwise $\text{result}(c_1, c_2, x) = \perp$.

At this point the function `complete` can be defined, that results in a minimal set of inequalities which is closed.

Definition 14. $I \cong I'$ iff for all $Y \subseteq X$ such that $|Y| \leq 2$, $\pi_Y(I) \equiv \pi_Y(I')$.

Definition 15. Let $I \subseteq \text{TVPI}_X$. Put $I_0 = \text{canon}(I)$ and $I_{i+1} = \text{canon}(I_i \cup \text{result}(I_i, I_i))$. Then `complete` : $\mathcal{P}(\text{TVPI}_X) \rightarrow \mathcal{P}(\text{TVPI}_X)$ is defined

$$\text{complete}(I) = I_n \text{ where } I_{n+1} \cong I_n \text{ and for every } 0 \leq m < n, I_{m+1} \not\cong I_m.$$

The following result (proved in [6]) states that when adding a single new inequality to a complete system (a system whose completion is itself) at most two `result` steps are needed to generate a new complete system.

Theorem 1. [6] Consider adding $c_0 \in \text{TVPI}_X$ to $I \subseteq \text{TVPI}_X$ where $\text{complete}(I) = I$. If $c \in \text{complete}(I \cup \{c_0\})$ and $c \neq \text{false}$, then one of the following holds:

1. $c \in I \cup \{c_0\}$
2. $c = \text{result}(c_0, c_1)$ where $c_1 \in I$
3. $c = \text{result}(\text{result}(c_0, c_1), c_2)$ where $c_1, c_2 \in I$

4.2 A Problem

The problematic interaction between closure and the classic widening is illustrated for Octagons by Miné [5], and has been reiterated in [28]. Consider the following inequalities over $\text{Oct}_{\{x,y,z\}}$, where iterations are increments of i .

$$C_i = \{y - x \leq i + 1, x - y \leq i + 1, z - x \leq i + 1, x - z \leq i + 1, y - z \leq 1, z - y \leq 1\}$$

Suppose an initial abstraction is:

$$A = \{y - x \leq 1, x - y \leq 1, y - z \leq 1, z - y \leq 1\}$$

then

$$A_0 = \text{complete}(A) = \{y - x \leq 1, x - y \leq 1, z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

Now consider the first iteration without using widening:

$$A_1 = A_0 \sqcup C_0 = \{y - x \leq 1, x - y \leq 1, z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

The next iteration is calculated with widening, where widening drops the unstable constraints on x, y , but closure using the `complete` function immediately introduces

new constraints on these variables:

$$A_1 \sqcup C_1 = \{y - x \leq 2, x - y \leq 2, z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

$$A_1 \nabla C_1 = \{z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

$$A_2 = \text{complete}(A_1 \nabla C_1) = \{y - x \leq 3, x - y \leq 3, z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

The next iterate is then the below, noting that this time it is the constraints on x, z that are unstable:

$$A_3 = \text{complete}(A_2 \nabla C_2) = \{y - x \leq 3, x - y \leq 3, z - x \leq 4, x - z \leq 4, y - z \leq 1, z - y \leq 1\}$$

And so on. Closure has not been accounted for in the use of stability in this definition of a widening, and this is not a terminating sequence. Therefore, ∇ is not a widening in this context. The solution proposed is simple [5]. At the widening point retain the widened ($A_i \nabla C_i$, not closed) version of the Octagon and use this as the first argument of the classic widening in the next iteration. A closed version of this may be used for continuing the abstract computation.

Gange, et al [28] isolate the problem: lattices are semantic, classic widening is syntactic, and closure has no semantic effect (at least when considering the system as a whole). But this representation dependence of the correct behaviour is “not fully satisfactory” [5].

The solution presented in [14, 28], introduces a new concept of isolated widening operating on an alternative structure. In [29] elements of weakly relational domains are considered purely geometrically [30], allowing the classic approach to widening to be retained, sidestepping the syntactic problems of closure. Their work also leads to a closure algorithm for Octagons, so that other domain operations can still use this form.

4.3 Entailment and Widening

As discussed in the introduction, polyhedra can be thought of in two ways: as convex sets of points, with this providing a semantics, or as the representation of these points, here this syntax is sets of linear constraints. To put it another way, is the lattice of polyhedra a lattice of convex spaces, or is it a lattice of sets of inequalities? Continuing to analyse the example above, the problem arises because widening is essentially a semantic notion about how (for geometric systems) the space is allowed to expand, but that it is often implemented syntactically by observing the form of constraints. Closure is a syntactic notion and the redundant inequalities introduced interfere with the straightforward correspondence, that the classic approach to widening relies upon, between stability of inequalities and the way in which they are removed.

A widening is given below that injects some semantics into the procedure, using entailment checking to determine whether or not apparently stable inequalities should be retained.

Suppose that A_i is a closed system of TVPI inequalities, and that widening is to be applied together with C_i to give the next iterate A_{i+1} . As usual, widening takes place between A_i and $A_i \sqcup C_i$. The widening proposed performs the classic widening on these

inputs, closes the system, then uses entailment checking to remove inequalities whose apparent stability is an artefact of the closed representation maintaining redundant inequalities.

Firstly, compute the classic widening (which of course is not a widening in this context, as discussed in Section 4.2), giving a set of constraints:

$$\text{complete}(A_i \nabla (A_i \sqcup C_i)) = \{c_1, \dots, c_k, c_{k+1}, \dots, c_n\}$$

Here, $c_1, \dots, c_k \notin A_i$ (that is, are introduced by closure) and $c_{k+1}, \dots, c_n \in A_i$. Then the new widening $A_{i+1} = A_i \nabla' C_i$ is given by:

$$A_{i+1} = A_i \nabla' C_i = \left\{ c_j \left| \begin{array}{l} k+1 \leq j \leq n, A_i \setminus \{c_j\} \not\models c_j \\ \text{or } A_i \setminus \{c_j\} \models c_j \text{ and } c_{k+1}, \dots, c_{j-1}, c_{j+1}, \dots, c_n \models c_j \end{array} \right. \right\}$$

That is, stable inequalities are dropped if $A_i \setminus \{c_j\} \models c_j$ (they are redundant in A_i) and $c_{k+1}, \dots, c_{j-1}, c_{j+1}, \dots, c_n \not\models c_j$ (they are non-redundant in $\text{complete}(A_i \sqcup C_i)$). That is, an inequality is not allowed to change its status from being a syntactic artefact of a closed system whose removal has no semantic impact, to an inequality whose removal will have a semantic impact, changing the convex space described.

Lemma 1. Where $A_{i+1} = A_i \nabla' C_i$ for closed A_i , $\text{complete}(A_{i+1}) = A_{i+1}$

Proof. Suppose some $c_\ell \in \text{complete}(A_{i+1})$, but $c_\ell \notin A_{i+1}$. Then c_ℓ can be derived from inequalities in A_{i+1} using the result operation. Since $A_{i+1} \subseteq A_i$, c_ℓ can also be derived from A_i , hence $c_\ell \in A_{i+1}$ by definition. \square

Lemma 2. ∇' is a widening.

Proof. Since ∇' only retains inequalities explicit in the previous iterations, and all iterates are finite sets, stability is reached for a set of constraints or the empty set. \square

Returning to the example from Section 4.2. A_0 is as before:

$$A_0 = \text{complete}(A) = \{y - x \leq 1, x - y \leq 1, z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

Next, A_1 is again as before, with no widening applied:

$$A_1 = \{y - x \leq 1, x - y \leq 1, z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

Now compute the classic widening. Here, the constraints in x, y are unstable, but new constraints on these variables are introduced in closure.

$$A_1 \sqcup C_1 = \{y - x \leq 2, x - y \leq 2, z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

$$\text{complete}(A_1 \nabla (A_1 \sqcup C_1)) = \{y - x \leq 3, x - y \leq 3, z - x \leq 2, x - z \leq 2, y - z \leq 1, z - y \leq 1\}$$

Now compute A_2 and observe that

$$A_2 = A_1 \nabla' C_1 = \{y - z \leq 1, z - y \leq 1\}$$

The inequalities in x, y are not included in A_1 , hence are not stable and are dropped. The inequalities in x, z are stable, but they were redundant in A_1 (for example, $x - y \leq 1, y - z \leq 1 \models x - z \leq 2$), and are not entailed by the other stable inequalities in $\text{complete}(A_1 \nabla (A_1 \sqcup C_1))$, hence are also dropped. Observe that $\text{complete}(A_2) = A_2$, and that this is a fixpoint.

Is this fully satisfactory? Not necessarily, the extensive use of entailment checking might make this widening particularly expensive to apply. However, it might be that a practical implementation marks inequalities introduced by closure, which fits with the observation that maintaining the widening point as a non-closed system allows a fixpoint to be observed.

The tension between the semantic notion of widening and the syntactic might not be problematic when the convex space has a unique syntactic representation. However, at least for constraint based systems, a potential problem exists even if a closed system is not maintained: suppose that a point is to be represented as a polyhedron over multiple variables; which constraints are used to do this, there are infinitely many choices? Two successive iterates need to choose the same representation for stability to be syntactically observed. It might be hoped, indeed expected, that this will be the case, but this needs to be enforced.

A final problem remains. The new widening results in a closed system, but part of the working involves applying closure to the system resulting from the classic widening. This potentially means calculating closure from scratch, which is an expensive operation; what refinements are available to avoid the complete recalculation of closure?

5 Linear Widening

Section 4.3 gave a widening operator that preserves canonical form for TVPI systems. However, widening can be more intricate than this, for example, as discussed in Section 2 thresholds might be used to weaken inequalities and observe whether the resulting weakened system is a fixpoint. This section proposes a more sophisticated widening which can perform this kind of reasoning for weakly relational domains.

Weakly relational domains can be split into two categories: those with a bounded number of independent inequalities such as Int_X , DBM_X and Oct_X , and those with a potentially unbounded number such as Log_X , TVPI_X and Poly_X . The widening problems for these two are subtly different. If the number of inequalities is not bounded, then widening must determine the number of inequalities in the fixpoint. If the number of inequalities is bounded, then the problem can be reduced to finding the constants on the right hand side of each inequality. This can be seen as finding the shape and finding the scaling of the inequalities that overapproximate the fixpoint.

One way to implement this approach is to use templates. The templates approach to polyhedra-based static analysis discovers potential inequality templates from the

code before performing a fixpoint calculation [31, 32]. The idea is that the relationships between variables can be found by a pass through a program, giving the coefficients of the variables in an inequality (not necessarily two variable inequalities) and the fixpoint calculation is performed with respect to these inequalities rather than arbitrary polyhedra. That is, the fixpoint is searching for the constant for an inequality template.

This emphasises that a fixpoint calculation is searching for two things. Firstly the template of the inequalities in a fixpoint (that is, the a, b for $ax + by \leq e$), secondly the constant for that inequality. The fixpoint calculation might be viewed as having two stages, the first to discover potential templates, the second to discover a constant that means that the inequality is part of a fixpoint. A special case of this is for unary inequalities, where finding a bound on a variable is particularly important, and in the case of TVPI in canonical form it is known that these bounds will be present. This section focuses on the case where the potential templates have been discovered, but the constants are not yet known.

5.1 Relational Templates for Widening

Let $\mathbb{Q}_\infty = \mathbb{Q} \cup \{\infty\}$ with $\forall e \in \mathbb{Q}. e < \infty$. If t is a linear expression over variables in X and constants in \mathbb{Q} then a *template constraint* is a map: $S_t : \mathbb{Q}_\infty \rightarrow \text{Poly}_X$ given by:

$$\begin{aligned} S_t(e) &= t \leq e \\ S_t(\infty) &= 0 * x \leq 0 \end{aligned}$$

For example, if $x, y \in X$ then $S_{3x-2y}(7) = 3x - 2y \leq 7$. Given a set of template constraints $S_{t_1} \dots S_{t_n}$ a *template* $T : \mathbb{Q}_\infty^n \rightarrow \mathcal{P}(\text{Poly}_X)$ is formed by:

$$T(\langle e_1, e_2, \dots, e_n \rangle) = \bigcup_{i \in [1, n]} S_{t_i}(e_i)$$

For example, using $C_x, C_{-x}, C_{x-y}, C_{y-x}$ gives a template such that $T(\langle 1, 2, 3, 4 \rangle) = \{x \leq 1, -x \leq 2, x - y \leq 3, y - x \leq 4\}$.

By careful selection of the template constraints, templates can be produced where $T(\mathbb{Q}_\infty^n)$ is contained in TVPI_X , Log_X , Oct_X or any other subset. There are many possible techniques for generating templates. For Oct_X , DBM_X and Int_X there is only a finite set of valid linear combinations over X so an obvious template is the one which contains all of them. Another approach is to convert a set of constraints into a template by converting the constants on the right hand side of the inequalities into the template parameters. One way to do this is to analyse the loop for a fixed number of iterations or until the shape of the constraints stabilises and then convert into a template.

5.2 Linear Widening and Fixpoint Prediction

The goal is to find template parameters such that a fixpoint is reached. This is done by making predictions as to these values based on observing a sequence of iterations.

This section assumes that *sequence predictors* p_i are available which take a finite prefix of a sequence of monotonic numbers in \mathbb{Q}_∞ and return a prediction of the fixpoint of the sequence. In the general case it is impossible to consistently predict the fixpoint as there will always be an infinite number of sequences with the same prefix and different fixpoints. However there are a number of cases in which predictions are possible because real-world programs often produce predictable sequences. Thresholds [19] are one example of this kind of prediction. Fitting geometric or arithmetic sequences and using their closed form and convergence results is another. Interpolating a polynomial and finding its maximum would be another possibility. Combinations of these taking the minimum predicted fixpoint would give a more robust approach.

To calculate a fixpoint, a number of iterations of the fixpoint calculations will be observed, and for each template constraint the sequence of constants noted. Using this sequence of integers to predict a template constraint that forms part of the fixpoint is referred to as *linear widening*. If A_0 is the abstract domain value before the fixpoint calculation and the abstract interpretation of one iteration of the fixpoint can be represented as $L : \mathcal{P}(\text{Poly}_X) \rightarrow \mathcal{P}(\text{Poly}_X)$ then linear widening can be performed using a template T (which can be computed after the initial observations) and a set of sequence predictors p_i . First observe α iterations of the loop:

$$A_{i+1} = L(A_i) \quad i < \alpha$$

Record the abstract domains computed and use them to find $v_i \in \mathbb{Q}_\infty^n$, the vector of parameters for the template, which contain each domain:

$$A_i \sqsubseteq T(v_i) \quad i \geq 0$$

Any technique that computes enclosing v_i will work. However it is desirable that the technique minimises the overapproximation of $T(v_i)$. If the A_i are in canonical form, computing v_i can be a simple look up of the relevant constraint. More complex entailment checks can give more tightly enclosing v_i . Where π_k selects the k^{th} component of a vector and $s_{i,j} \in \mathbb{Q}_\infty$, these vectors form n sequence prefixes $s_{0,k}, \dots, s_{i+1,k}$ by taking the k^{th} component of each vector:

$$\begin{aligned} s_{0,k} &= \pi_k(v_0) \\ s_{i+1,k} &= \max(s_{i,k}, \pi_k(v_{i+1})) \end{aligned}$$

Using the sequence predictors p_i on each of the component sequences forms a vector $u_{\alpha+j}$ of predicted bounds on sequence fixpoints:

$$u_{\alpha+j} = (p_1(s_{0,1}, \dots, s_{\alpha+j,1}), \dots, p_n(s_{0,n}, \dots, s_{\alpha+j,n}))$$

The template is used to turn predicted fixpoints on \mathbb{Q}_∞^n into a predicted fixpoint on $\mathcal{P}(\text{Poly}_x)$ which is then used to compute the next iteration of the loop:

$$A_{\alpha+j+1} = L(\text{complete}(T(u_{\alpha+j}))) \quad j < \beta$$

Finally, the widening from Section 4.3 is used to compute a fixpoint if one has not been reached:

$$A_{\alpha+\beta+1} = A_{\alpha+\beta-1} \nabla' A_{\alpha+\beta}$$

Although this procedure is heuristic and is heavily dependent on the sequence predictors, it can be surprisingly effective as demonstrated in Section 6.1.

6 Discussion with Worked Examples

6.1 Widening to a Limit

Consider a transfer function for sets C_i . This translates a shape (approximation) by one unit in the y -axis, and takes its join with a point half way between the top right hand corner of the translated shape and $x = 2$, and with the original C_i .

$$C_{i+1} = C_i \cup \{(x, y + 1) \mid (x, y) \in C_i\} \cup \left\{ \left(1 + \frac{\max_x(C_i)}{2}, \max_y(C_i) + 1 \right) \right\}$$

With initial set $C_0 = \{(x, y) \mid -x \leq 0, y \leq 1, x - y \leq 0\}$, this gives the monotonic sequence of spaces illustrated in Figure 6. Loops with a single counter and an if statement could produce sequences of this form.

Consider computing a fixpoint for this sequence by applying widening. Analysing with $\text{TVPI}_{\{x,y\}}$ (note that these constraints are all in $\text{Log}_{\{x,y\}}$), each space can be precisely described and the set of constraints will grow as in the figure. Symbolically in canonical form the first three iterates are:

- $\text{canon}(C_0) = \{-x \leq 0, -y \leq 0, x \leq 1, y \leq 1, x - y \leq 0\}$
- $\text{canon}(C_1) = \{-x \leq 0, -y \leq 0, x \leq \frac{3}{2}, y \leq 2, x - y \leq 0, 2x - y \leq 1\}$
- $\text{canon}(C_2) = \{-x \leq 0, -y \leq 0, x \leq \frac{7}{4}, y \leq 3, x - y \leq 0, 2x - y \leq 1, 4x - y \leq 4\}$

Consider these in terms of the linear widening discussed in Section 5. The initial α iterates find a set of templates to widen to. A number of these can already be observed to be stable and these will be retained in the widened system. Observe that the upper bound on x is explicit because the iterates are maintained in canonical form, and that these bounds form a geometric sequence $1, \frac{3}{2}, \frac{7}{4}, \dots$. From this, the sequence predictor can deduce a limit of 2, hence the bound on x can be widening to $x \leq 2$. However, the bound on y follows an arithmetic sequence, $1, 2, 3, \dots$ and this does not suggest a value to widen to. In the β attempts to find a fixpoint some threshold values might be explored, but as fixpoints have no upper bound for y this constraint will be dropped. A fixpoint is found that over-approximates the space described by the union of all the iterates. The precise fixpoint depends on α the number of iterations applied before widening. To summarise, suppose that $\alpha = 3$ (so the last iterate before widening is C_3), then the fixpoint found is $\{-x \leq 0, -y \leq 0, x \leq 2, x - y \leq 0, 2x - y \leq 1, 4x - y \leq 4, 8x - y \leq 11\}$.

If a similar analysis were conducted with $\text{Int}_{\{x,y\}}$ or $\text{Oct}_{\{x,y\}}$ as the abstract domain, then linear widening is applied in the same way, again finding the upper bounds on x . However, the result would be greater over-approximations, since intervals can only describe the bounded box, and Octagons can only additionally describe

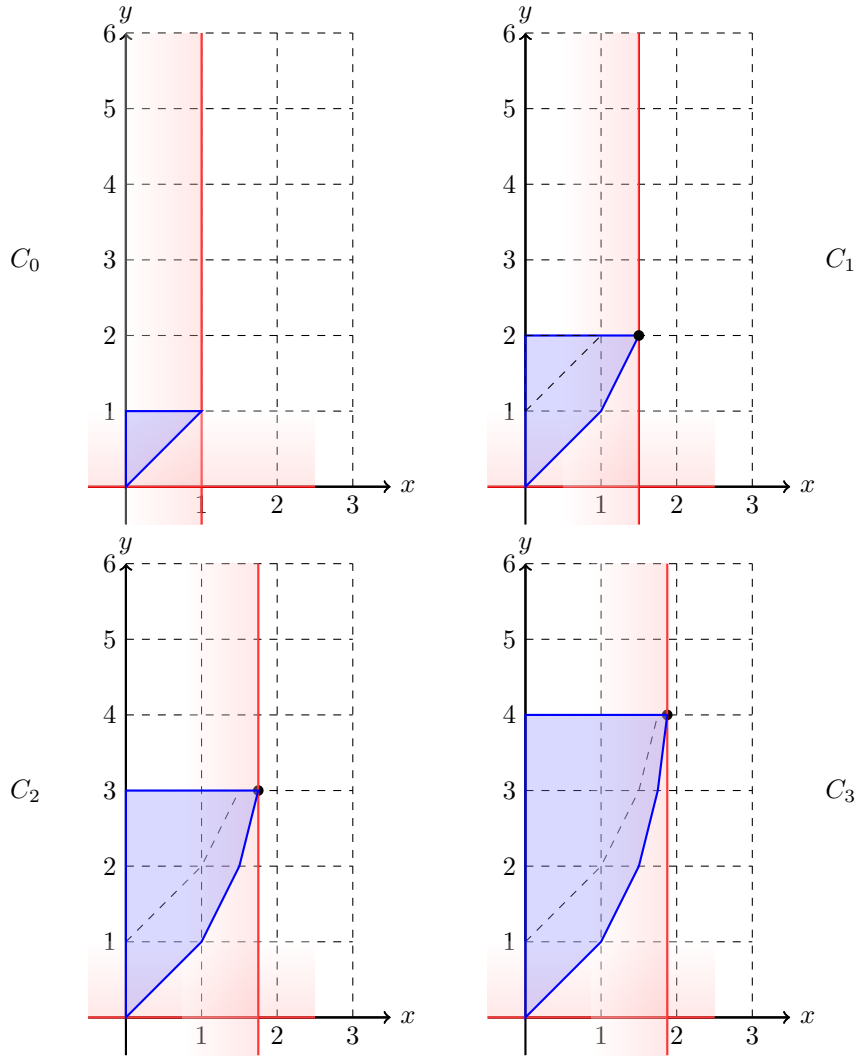


Fig. 6 A widening tribute to Zeno of Elea

the $x - y \leq 0$ constraint. Without the linear widening, the upper bound on x might still be found using a threshold, but only if the correct threshold is supplied up front.

The classic widening is typically applied after two or three iterations without widening. In this example, there is an attraction in applying more iterations to get a better approximation, since additional templates emerge at every iteration; this poses the question of what the right value for α is. The sequence of upper bounds for y does not lead to a fixpoint, but some potential bounds might be investigated; this poses the question of what the right value for β is. Finally, the analysis above requires the limit suggested by the sequence of upper bounds for x to be found, posing the question as to how this is integrated into an analysis framework.

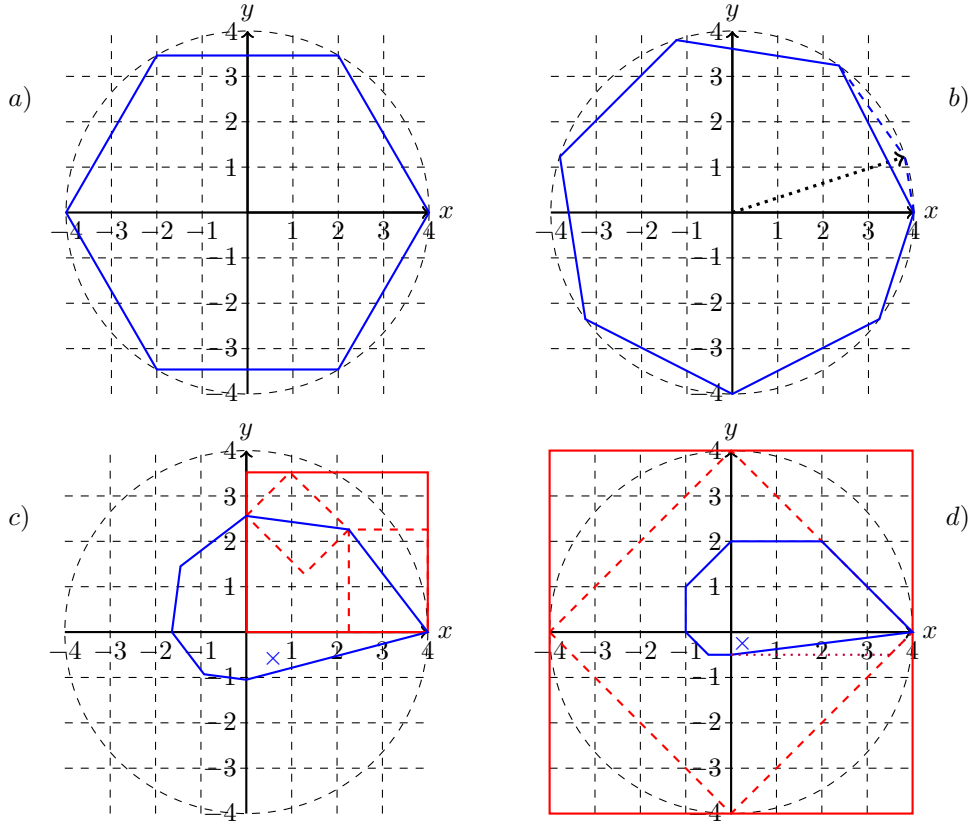


Fig. 7 Circle widening problem

6.2 Fixpoints and Rotations

Scientific programming often uses trigonometric functions. Analysis of these can be problematic, even given constant arguments. Consider a transfer function for set of points C_i , parameterised by constants $r \in \mathbb{R}$, and $\theta \in [0, 2\pi)$, an angle.

$$C_{i+1} = \{(r(x\cos(\theta) - y\sin(\theta)), r(x\sin(\theta) + y\cos(\theta)) \mid (x, y) \in C_i\} \sqcup C_i$$

This rotates the current iteration by θ , scaling by r and taking the least upper bound with the previous iterate. Notice that if $r = 1$ then vertices of the described space are points on the circumference of a circle; if $r > 1$ then points diverge; if $r < 1$ then points become closer to the centre of the circle. Note that the use of reals in this example means that coefficients for constraints might also be reals. Figure 7 illustrates some possible iterations with $C_0 = \{(4, 0)\}$, and various choices for θ and r .

Case a) has $\theta = \frac{\pi}{3}$ and $r = 1$. Here, a fixpoint describing the space is a hexagon which can be described with TVPI inequalities; this would be found assuming that the analysis was run for a sufficient number of iterations without widening. Notice that if the abstract domain used were intervals or Octagons then each iterate would

over-approximate the space, this space will diverge, and when widening is applied to enforce termination the result would be the top element, the whole plane.

Case b) has $\theta = \frac{3\pi}{10}$ and $r = 1$. This is somewhat similar to case a), although the analysis needs more iterations (and three times round the circle) to reach a fixpoint, which will be an icosagon. The diagram illustrates the approximation after seven iterations, with the arrow pointing to the point that will need to be incorporated when the next least upper bound is calculated. It is not too hard to see how θ can be chosen so that the fixpoint is arbitrarily complicated, or indeed non-existent. This might motivate some kind of relaxation, however, such a relaxation will always lead to an over-approximation of the space including points from without the circle, and the space diverges, as above. The desired fixpoint is the circle, but this cannot be represented using polyhedral domains.

As noted above, if $r > 1$ then the approximation will always diverges, hence the only fixpoint is the whole plane.

Now suppose that $r < 1$. There is a critical value for r where fixpoint behaviour changes, that is when $r = \frac{\sqrt{2}}{2}$. As can be seen in Figure 7 d) this is the value at which the iteration of a bounding box is contained within itself; this will be discussed further below. In Figure 7 c), $\theta = \frac{\pi}{4}$ and $r = 0.8 > \frac{\sqrt{2}}{2}$. The iterations are approximated by a spiral, leading to the fixpoint illustrated. Again, this can be modelled with TVPI constraints. Suppose instead that this was approximated using $\text{Int}_{\{x,y\}}$. C_1 would be overapproximated with the dashed box whose bottom right corner is $(4, 0)$. To calculate C_2 this space is translated to give the dashed box tilted by $\frac{\pi}{4}$ and the least upper bound is calculated, resulting in the containing box. C_2 contains points outside of the circle and further iterations will diverge. Once widening is applied the result will again be the plane, the top element of the lattice of $\text{Int}_{\{x,y\}}$.

Finally, consider the case illustrated in Figure 7 d). Here, $\theta = \frac{\pi}{4}$ and $r = \frac{\sqrt{2}}{2}$. To illustrate the point, a rather gross over-approximation of the space is given by the bounding box which translates into a subspace of itself at the next iteration, and widening need not be considered. As previously, this space can be described precisely in $\text{TVPI}_{\{x,y\}}$, whilst (as shown by the dotted lines) $\text{Oct}_{\{x,y\}}$ can also approximate this space, reaching a fixpoint.

This discussion has illustrated some of the issues arising when computing fixpoints of iterates built using trigonometric functions. It poses several questions: i) what is the correct patience to show before performing widening? ii) when and how can polyhedral approximations be relaxed to given improved performance and fixpoints? iii) what is the right domain to use when analysing code using trigonometric (as in this case, matrix) functions? One response to the last question might be that polyhedral approximations are the wrong choice and a domain capturing circles and ellipses, such as that in [33], might be a better choice. In this case, an additional question is how do polyhedral abstract domains interact with non-polyhedral abstract domains?

7 Conclusion

This paper has looked at the representation of weakly relational domains (TVPI in particular), widening for polyhedral constraints, and the interaction between the two.

It has recognised the important of maintaining two variables constraint systems in a canonical form, and has provided one such form. Widening has been discussed from a number of viewpoints. The interaction between widening and the closed forms maintained in weakly relational domains has been investigated and in Section 4.3 a new widening has been introduced that gives a new way to sidestep the problem. The linear widening problem (widening a series of iterates of an inequality template along a vector) has been defined and discussed. The application of these widenings alongside the maintenance of the canonical form is illustrated, and the paper concludes with a discussion of fixpoints for functions that rotate spaces. Future work includes empirical investigation of the application of the techniques discussed, and theoretical development of the interaction of the canonical form with other domain operations.

Acknowledgements

The authors would like to thank Marius Zicius for discussions on two variable per inequality constraints, and the anonymous referees for their insightful feedback. Research reported in this publication was partially supported by an Amazon Research Award, Fall 2022 CFP. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of Amazon.

References

- [1] Cousot, P., Cousot, R.: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: Conference Record of the Fourth ACM Symposium on Principles of Program Analysis, pp. 238–252. ACM Press, New York (1977)
- [2] Cousot, P.: Principles of Abstract Interpretation. MIT Press, Cambridge, Massachusetts (2021)
- [3] Cousot, P., Halbwachs, N.: Automatic Discovery of Linear Restraints among Variables of a Program. In: Principles of Programming Languages, pp. 84–97. ACM Press, New York (1978)
- [4] Simon, A., King, A., Howe, J.M.: Two Variables per Linear Inequality as an Abstract Domain. In: Logic Based Program Development and Transformation. Lecture Notes in Computer Science, vol. 2664, pp. 71–89. Springer, Berlin (2002)
- [5] Miné, A.: The Octagon Abstract Domain. Higher-Order and Symbolic Computation **19**(1), 31–100 (2006)
- [6] Howe, J.M., King, A., Simon, A.: Incremental Closure for Systems of Two Variables Per Inequality. Theoretical Computer Science **768**, 1–42 (2019)
- [7] Brain, M., Howe, J.M.: Widening for Systems of Two Variables Per Inequality. In: Proceedings of the 8th SC-Square Workshop. CEUR Workshop Proceedings,

vol. 3455, pp. 11–28 (2023)

- [8] Monniaux, D.: A minimalistic look at widening operators. *Higher-Order and Symbolic Computation* **22**, 145–154 (2009)
- [9] Monniaux, D.: On the decidability of the existence of polyhedral invariants in transition systems. *Acta Informatica* **56**(4), 385–389 (2019)
- [10] Howe, J.M., King, A.: Logahedra: a New Weakly Relational Domain. In: *International Symposium on Automated Technology for Verification and Analysis. Lecture Notes in Computer Science*, vol. 5799, pp. 306–320. Springer, Berlin (2009)
- [11] Reinbacher, T., Függer, M., Brauer, J.: Runtime verification of embedded real-time systems. *Formal Methods in System Design* **44**, 203–239 (2014)
- [12] Cousot, P., Cousot, R., Feret, J., Miné, A., Rival, X.: Why does ASTRÉE scale up? *Formal Methods in System Design* **35**, 229–264 (2009)
- [13] Miné, A.: A New Numerical Abstract Domain Based on Difference-Bound Matrices. In: *Second Symposium on Programs as Data Objects. Lecture Notes in Computer Science*, vol. 2053, pp. 155–172. Springer, Berlin (2001)
- [14] Gange, G., Ma, Z., Navas, J.A., Schachte, P., Søndergaard, H., Stuckey, P.: A Fresh Look at Zones and Octagons. *ACM Transactions on Programming Languages and Systems* **43**(3), 1–51 (2021)
- [15] Cousot, P., Cousot, R.: Comparing the Galois Connection and Widening/-Narrowing Approaches to Abstract Interpretation. In: *Proceedings of the 4th International Symposium on Programming Language Implementation and Logic Programming. Lecture Notes in Computer Science*, vol. 631, pp. 269–295. Springer, Berlin (1992)
- [16] Bourdoncle, F.: Efficient chaotic iteration strategies with widenings. In: *Proceedings of the International Conference on Formal Methods in Programming and Their Applications. Lecture Notes in Computer Science*, vol. 735, pp. 128–141. Springer, Berlin (1993)
- [17] Fukuda, K., Prodon, A.: Double Description Method Revisited. In: *Combinatorics and Computer Science. Lecture Notes in Computer Science*, vol. 1120, pp. 91–111. Springer, Berlin (1995)
- [18] Bagnara, R., Hill, P.M., Mazzi, E., Zaffanella, E.: Precise widening operators for convex polyhedra. *Science of Computer Programming* **58**(1–2), 28–56 (2005)
- [19] Blanchet, B., Cousot, P., Cousot, R., Feret, J., Mauborgne, Miné, A., Monniaux, D., Rival, X.: Design and Implementation of a Special-Purpose Static Program

- Analyzer for Safety-Critical Real-Time Embedded Software. In: *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*. Lecture Notes in Computer Science, vol. 2566, pp. 85–108. Springer, Berlin (2002)
- [20] Simon, A., King, A.: Widening Polyhedra with Landmarks. In: *Proceedings of the 4th Asian Symposium on Programming Languages and Systems*. Lecture Notes in Computer Science, vol. 4279, pp. 166–182. Springer, Berlin (2006)
- [21] Simon, A.: *Value-Range Analysis of C Programs*. Springer, Berlin (2008)
- [22] Gopan, D., Reps, T.W.: Lookahead Widening. In: *Proceedings of the 18th International Conference on Computer Aided Verification*. Lecture Notes in Computer Science, vol. 4144, pp. 452–466. Springer, Berlin (2006)
- [23] Simon, A., Chen, L.: Simple and precise widenings for polyhedra. In: *Proceedings of the 8th Asian Symposium on Programming Languages and Systems*. Lecture Notes in Computer Science, vol. 6461, pp. 139–155. Springer, Berlin (2010)
- [24] Gonnord, L., Halbwachs, N.: Combining Widening and Acceleration in Linear Relation Analysis. In: *Proceedings of the 13th International Static Analysis Symposium*. Lecture Notes in Computer Science, vol. 4134, pp. 144–160. Springer, Berlin (2006)
- [25] Brain, M., Joshi, S., Kroening, D., Schrammel, P.: Safety Verification and Refutation by k -Invariants and k -Induction. In: *Proceedings of the 22th International Static Analysis Symposium*. Lecture Notes in Computer Science, vol. 9291, pp. 145–161. Springer, Berlin (2015)
- [26] Zaffanella, E., Arceri, V.: “Fixing” the Specification of Widenings. In: Arceri, V., Cortesi, A., Ferrara, P., Olliaro, M. (eds.) *Challenges of Software Verification*. Intelligent Systems Reference Library, vol. 238, pp. 57–76. Springer, Berlin (2023)
- [27] Ballou, K., Sherman, E.: Incremental Transitive Closure for Zonal Abstract Domain. In: *Proceedings of the 14th International Symposium NASA Formal Methods*. Lecture Notes in Computer Science, vol. 13260, pp. 800–808. Springer, Berlin (2022)
- [28] Gange, G., Navas, A. J., Schachte, P., Søndergaard, H., Stuckey, P.: Dissecting widening: Separating termination from information. In: *Proceedings of the 17th Asian Symposium on Programming Languages and Systems*. Lecture Notes in Computer Science, vol. 11893, pp. 95–114. Springer, Berlin (2019)
- [29] Bagnara, R., Hill, P.M., Mazzi, E., Zaffanella, E.: Widening operators for weakly-relational numeric abstractions. In: *Proceedings of the 12th International Symposium on Static Analysis*. Lecture Notes in Computer Science, vol. 3672, pp. 3–18. Springer, Berlin (2005)

- [30] Bagnara, R., Hill, P.M., Zaffanella, E.: Weakly-relational Shapes for Numeric Abstractions: Improved Algorithms and Proofs of Correctness. *Formal Methods in System Design* **35**(3), 279–323 (2009)
- [31] Sankaranarayanan, S., Dang, T., Ivančić, F.: Symbolic Model Checking of Hybrid Systems Using Template Polyhedra. In: *Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science*, vol. 4963, pp. 188–202. Springer, Berlin (2008)
- [32] Sankaranarayanan, S., Colón, M., Sipma, H.B., Manna, Z.: Efficient Strongly Relational Polyhedral Analysis. In: *Verification, Model Checking, and Abstract Interpretation. Lecture Notes in Computer Science*, vol. 3855, pp. 111–125. Springer, Berlin (2006)
- [33] Feret, J.: Static Analysis of Digital Filters. In: *Proceedings of the 13th European Symposium on Programming. Lecture Notes in Computer Science*, vol. 2986, pp. 33–48. Springer, Berlin (2004)