

# Widening for Systems of Two Variables Per Inequality

Martin Brain, Jacob M. Howe

*Department of Computer Science, City, University of London, EC1V 0HB, UK*

## Abstract

The abstract interpretation approach to program verification involves symbolically calculating fixpoints over lattices. Integral to these fixpoint calculations is an operation called widening, which discards information in order to move up the lattice so as to ensure termination, as well as potentially aiding efficiency. Abstract interpretation often uses lattices of numeric constraints where the number of variables allowed in a constraint is restricted, called weakly relational domains. This paper is concerned with the application of widening with weakly relational domains. One particular point of interest is the problematic interaction between widening and the maintenance of a closed form for weakly relational domains. The solution to this problem uses entailment, which essentially involves satisfiability checking for numeric constraints.

## Keywords

Abstract Interpretation, Weakly Relational Domains, Widening

## 1. Introduction

Abstract interpretation [1, 2] is an approach to formal program verification. 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. 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 (potentially) be an over-approximation of the least fixpoint. This fixpoint can then be interpreted back into the language of the program allowing verification conditions to be checked.

The abstract domain might be based on one of many things: sets, Boolean functions, numeric constraints, etc. In many cases, it is possible that the lattice for the abstract domain contains infinite ascending chains, hence termination of the fixpoint calculation is not guaranteed. Even if termination is guaranteed, long chains might mean that termination doesn't 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.

Abstract interpretation has proved to be one of the most scalable approaches to program verification. Key to this is being able to disregard connections between variables and treat them

---


*8th International Workshop on Satisfiability Checking and Symbolic Computation, July 28, 2023, Tromsø, Norway, Collocated with ISSAC 2023*

✉ martin.brain@city.ac.uk (M. Brain); j.m.howe@city.ac.uk (J. M. Howe)

🆔 0000-0003-4216-7151 (M. Brain); 0000-0001-8013-6941 (J. M. Howe)



© Copyright 2023 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

as independent. This is manifested in the representation, that is, the choice of abstract domain, then reflected in the way computations are modelled as transformers, as well as the use of widening in handling loops. Not tracking relationships in the abstract domain gives scalability but at the cost of precision; the results are over-approximate, hence the results are safe, but possibly giving false alarms. Too many false alarms makes the verification process unusable.

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 relation domains. These restrict the constraints in the abstract domain in some way, typically allowing at most two variables in each equality [4] and often also restricting the coefficients of the variables in the inequalities, for example to  $+1, 0, -1$  to give Octagons [5].

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, and no progress is made.

This paper is about widening, and especially about the geometric challenges arising when working with a (weakly) relational numeric domain to handle some of the relationships that occur between variables.

The paper makes the following contributions:

- provides a review of widening techniques for numeric domains and illustrates their application;
- gives case studies to illustrate problems with the application of widening;
- proposes a new way of treating widening for weakly relational domains maintained in a closed form.

The rest of the paper is structured as follows: Section 2 surveys related work and introduces relevant notation, Section 3 contains a worked example demonstrating the application of weakly relational domains and widening in a program verification context, Section 4 gives problems illustrating points of interest, and provides some tentative solutions, Section 5 concludes.

## 2. Background and Related Work

This work is concerned with widening and abstract domains consisting of sets of linear inequalities, or polyhedral domains. This section gives background on polyhedra and their subclasses, a brief history of widening, details on weakly relational domains and closure, before illustrating the problematic interaction between closure and widening.

### 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,  $\text{Poly}_X$ , the set of all linear inequalities over a set of variables  $X$ . Here (and throughout this paper), coefficients and constants are rationals, but other choices 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 a set of constraints over a subset of variables is an important operation in abstract interpretation.

Abstract interpretation using the  $\text{Poly}_X$  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. 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 [7] restricts coefficients to be powers of 2 (potentially with a maximum power). Structures based on Logahedra have been used as atomic structures for verification of embedded real-time systems [8].

**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 [9]. 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 [10, 11], 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  $\text{TVPI}_X$  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, 12] 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 then a function  $\nabla : L \times L \rightarrow L$  such that:

- i)  $\forall x, y \in L. x \sqsubseteq x \nabla y$
- ii)  $\forall x, y \in L. y \sqsubseteq x \nabla y$
- iii) 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 possible values of variables change 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 point 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 [13].

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 (which contains an implicit assumption of non-redundancy), but variations on this approach, whatever the representation, remain the classic approach. A number of heuristics as to how and when to widen are available and have been combined and refined in [14]. 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 can't verify a correctness condition. Various alternatives have been investigated. This includes widening with thresholds [15], where the user supplies, before analysis, a series of thresholds to widen to. So, for example, in the classic approach  $[1, 4] \nabla [1, 5] = [1, \infty]$ , but if the user has supplied some threshold  $n$ , 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; but, 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, that is, bounds on single variables, and not capturing relational constraints. Another approach is to automate the selection of thresholds, as suggested in the widening with landmarks approach of [16, 17]. The lookahead widening of [18] is another attempt to find good values to widen to. Work on widening continues, with [19] considering how it can be used when analysing programs containing non-linear transformations, [20] tackling loops and how their analysis can be used to aid the successful application of widening, whilst [21] provides a more general framework where abstract interpretation can interact with bounded model checking and k-induction.

### 2.3. Weakly Relational Domains and 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. 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.

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.

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 set of variables: simply select those inequalities including those variables.

**Definition 9.** The *syntactic projection*, denoted  $\pi_Y$  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. Notice that a closed system may include inequalities that are redundant even in a given projection (unary constraints). The entailment operation,  $\models$ , is geometric inclusion. If  $C_1 \models C_2$  and  $C_2 \models C_1$ , then  $C_1 \equiv C_2$ . The definition of closed states that syntactic and semantic projection coincide:

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

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

Some inequalities in a projection might not be wanted, since they are redundant with respect to all projections. The filter operation will be used to remove these, giving a set of inequalities that is a minimal representation equivalent to the projection (noting that unary constraints are retained, and that points and lines might have many such representations):

**Definition 11.** The mapping filter  $:\mathcal{P}(\text{TVPI}_X) \rightarrow \mathcal{P}(\text{TVPI}_X)$  is defined:

$$\text{filter}(C) = \cup\{\text{filter}_Y(\pi_Y(C)) \mid Y \subseteq X \wedge |Y| \leq 2\}$$

where:

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

The process of generating implied inequalities is based on the result operator that eliminates a shared variable from a pair of inequalities. In the definition of complete below, result will be lifting to sets of inequalities, referring to all pairwise applications of result.

**Definition 12.** 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 13.**  $I \cong I'$  iff for all  $Y \subseteq X$  such that  $|Y| \leq 2$ ,  $\pi_Y(I) \equiv \pi_Y(I')$ .

**Definition 14.** Let  $I \subseteq \text{TVPI}_X$ . Put  $I_0 = \text{filter}(I)$  and  $I_{i+1} = \text{filter}(I_i \cup \text{result}(I_i, I_i))$ . 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.** 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  $c_1, c_2 \in I$

## 2.4. Widening and Closure: A Problem

There is a problem when using widening whilst calculating a fixpoint over weakly relational domains. Widening attempt to weaken a system of inequalities by discarding inequalities that are not displaying stability. This is syntactically calculated: consider  $x$  and  $y$  as sets of inequalities, then  $x \nabla y = x \cap y$ . By maintaining a closed representation, 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.

This problem is illustrated for Octagons by Miné [5], and has been reiterated in [22]. 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  $x, z$ , 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 hasn't been accounted for in the use of stability in this definition of a widening, and this isn't a terminating sequence. Therefore,  $\nabla$  isn't 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 [22] 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 [22, 11], introduces a new concept of isolated widening operating on an alternative structure. In [23] elements of weakly relational domains are considered purely geometrically [24], allowed 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.

## 2.5. Relaxation

A final point of consideration is relaxation of systems of constraints. System of constraints  $y$  is said to be a relaxation of  $x$  if  $x \sqsubset y$ . Noting again that widening is about enforcing termination, it should also be noted that changing an element from an abstract domain might have another motivation, that is, to maintain a tractable size of representation in order that a fixpoint computation does not grind to a halt. The more expressive the set of constraints underlying the abstract domain is, the more this becomes an issue. The concept of relaxation is geometrical, but again interacts with the syntactic maintenance of a closed form.

## 3. 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  $j \in [0, n]$ . 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,  $i \in [0, 1], j \in [0, 1]$ . 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.

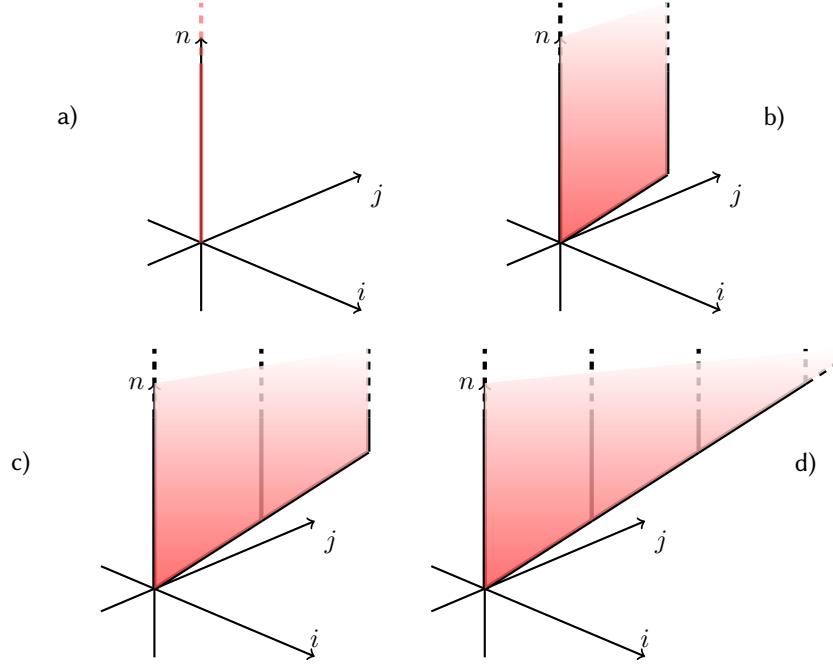
Repeating this will result in the intervals for the variables at line (8) increasing. However, this process will not terminate, since the values will not stabilise at a fixpoint. This necessitates widening. To accelerate (or indeed to reach) a fixpoint, information is discarded or generalised, until the abstract domain value at the merge point is stable. 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:

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

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  $\text{Int}$  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



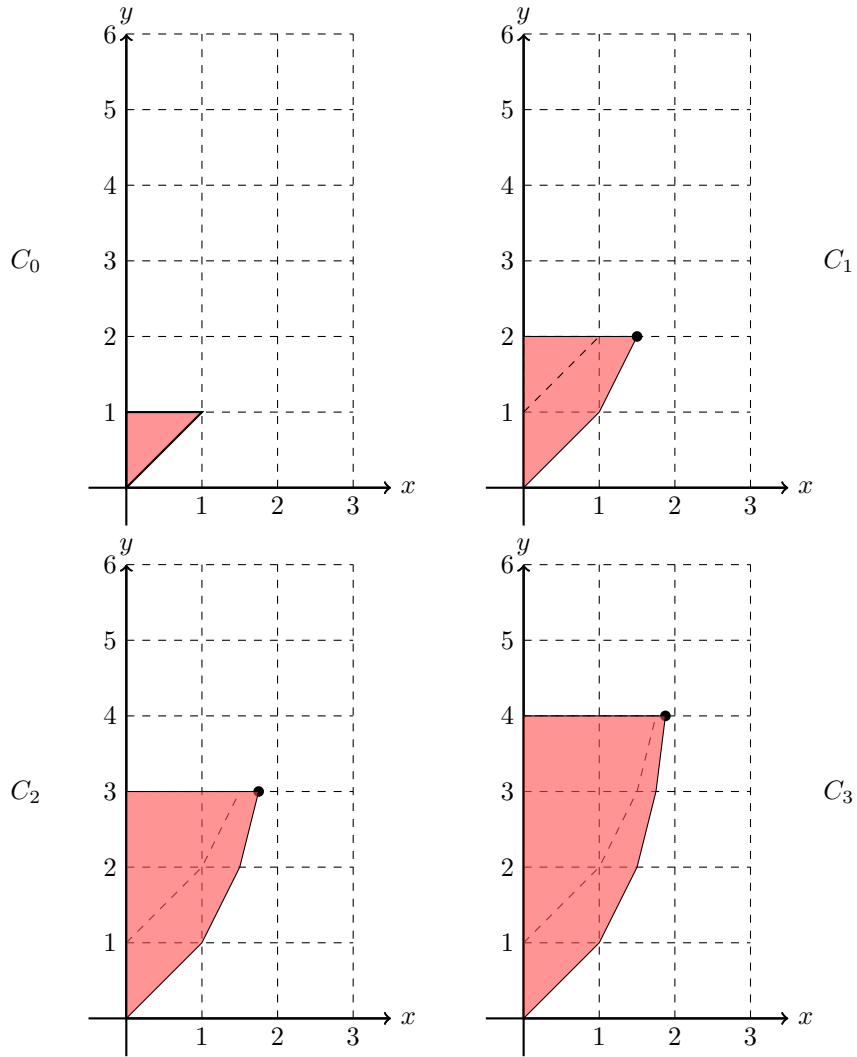
**Figure 1:** Fixpoint iterations over  $\text{TVPI}_{\{i,j,n\}}$ , with widening

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 results in  $\{-i \leq 0, -j \leq 0, i \leq n, i - j \leq 0, j - i \leq 0, i \leq 1, j \leq 1\}$ . This represents a section of plane reaching through  $i = j$  (see Figure 1 b)). Repeating this will result in the plane section increasing in size (as in Figure 1 c)). At the next iteration widening is applied, retaining only the stable inequalities  $\{-i \leq 0, -j \leq 0, i \leq n, i - j \leq 0, j - i \leq 0\}$ . as illustrated in Figure 1 d). This fixpoint implies that  $j \leq n$  (indeed in the closed representation used this will be 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.

## 4. Results

This section works through three examples illustrating the use of widening in over-approximating fixpoints within geometric structures. It discusses the related use of relaxation to throttle the size of representation and in particular addresses the problems that can arise whilst applying widening with weakly relational domains.



**Figure 2:** A widening tribute to Zeno of Elea

#### 4.1. Problem 1

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 2.

Consider applying widening to this sequence, with abstract domains  $\text{Int}_{\{x,y\}}$ ,  $\text{Oct}_{\{x,y\}}$ ,  $\text{TVPI}_{\{x,y\}}$ .

In the first instance, each  $C_i$  will be approximated by its bounding box. When classic widening is applied the increasing upper bounds on  $x$  and  $y$  will be unstable and the fixpoint  $\{-x \leq 0, -y \leq 0\}$  will be found. If widening with thresholds were applied and threshold  $t = 2$  were supplied then a stronger over-approximation would be found, namely  $\{-x \leq 0, x \leq t, -y \leq 0\}$ . Repeating with Octagons, with  $C_i$  for  $i \geq 1$ , the space can't be precisely described, and the end result after widening would add the  $x - y \leq 0$  constraint to the fixpoint calculated with intervals (either with or without thresholds).

Analysing with TVPI, each space can be precisely described and the set of constraints will grow as in the picture,  $\{-x \leq 0, -y \leq 0, x - y \leq 0, x - 2y \leq 0.5, x - 4y \leq 1, x - 8y \leq 1.5, \dots\}$  (plus upper bounds on  $x$  and  $y$ ). In fact, these constraints are in  $\text{Log}_{\{x,y\}}$ . When widening is applied, all but the final constraint (and the upper bounds) will be stable. Again, a widening with thresholds approach might add an upper bound on  $x$ .

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. This presents three problems to be addressed: i) the practical question of how many inequalities to maintain, and how to relax the system to control the number of inequalities; ii) the practical question of how to determine the number of iterations to apply before using widening; iii) the theoretical question of how to recognise the bound being approached as the limit of the sequence of inequalities, and add this without relying on up front thresholds.

## 4.2. Problem 2

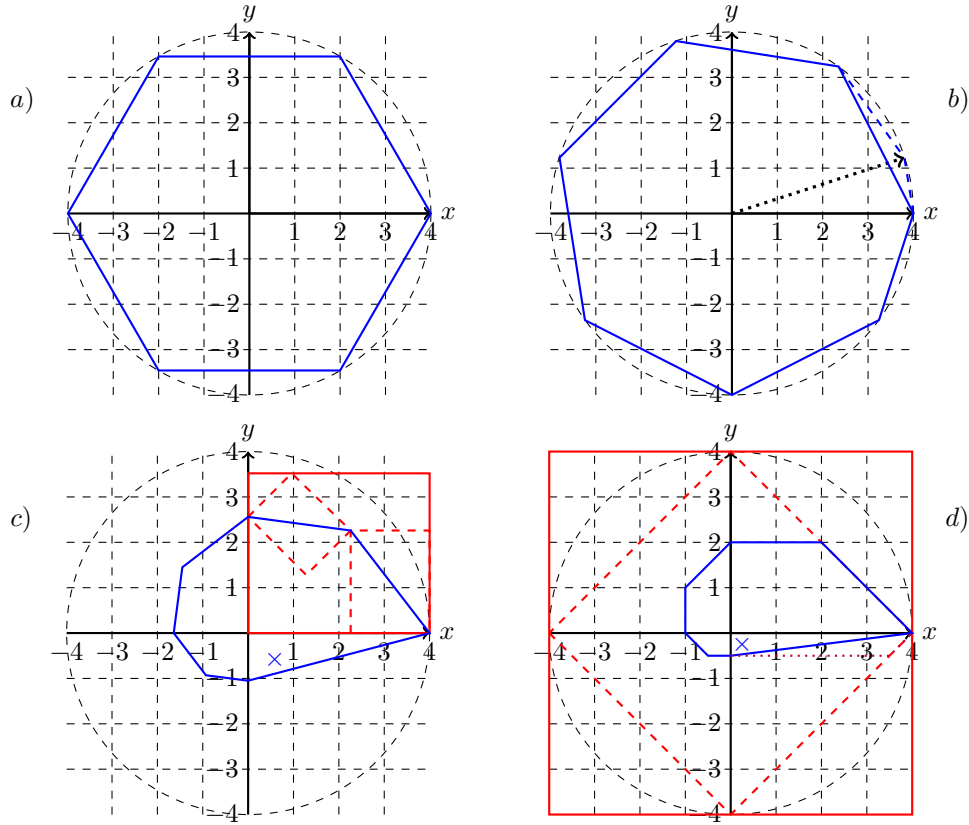
Scientific programming often uses trigonometric functions. Analysis of these can be problematic. Consider a transfer function for set of points  $C_i$ , parameterised by constant  $r$ , where  $\theta$  is an angle.

$$C_{i+1} = \{(r(x.\cos(\theta) - y.\sin(\theta)), r(x.\sin(\theta) + y.\cos(\theta)) \mid (x, y) \in C_i, \theta \in [0, 2\pi), r \in \mathbb{Q}\} \sqcup C_i$$

This rotates the current iteration by  $\theta$ , 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. Figure 3 illustrates some possible iterations with  $C_0 = \{(4, 0)\}$ , and various choices for  $\theta$  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 isn't



**Figure 3:** Circle widening problem

too hard to see how  $\theta$  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 can't be represented using polyhedral domains.

As noted above, if  $r > 1$  then the approximation will always diverge, 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 3 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 3 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 3 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 issue 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 [25], might be a better choice. In this case, an addition question is how do polyhedral abstract domains interact with non-polyhedral abstract domains?

### 4.3. Problem 3

This section returns to the example from Section 2.4, which demonstrated problematic behaviour when closure and widening interact. As noted there, 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. To put it another way, is the lattice of polyhedra a lattice of convex spaces, or is it a lattice of sets of inequalities? 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  $C_i$  are the inequalities being merged at a widening point. 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 isn't a widening in this context), giving a set of constraints  $c_\ell$ :

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

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:

$$A_{i+1} = \{c_j \mid k+1 \leq j \leq n, A_i \setminus \{c_j\} \not\models c_j, \text{ or } A_i \models c_j \text{ and } c_{k+1}, \dots, c_{j-1}, c_{j+1}, \dots, c_n \models c_j\}$$

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  $\text{complete}(A_i \sqcup C_i)$ ).

**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  $c_\ell$  can be derived from inequalities in  $A_{i+1}$  using the result operation. Since  $A_{i+1} \subseteq A_i$ ,  $c_\ell$  can also be derived from  $A_i$ , hence  $c_\ell \in A_{i+1}$  by definition.  $\square$

**Lemma 2.**  $\nabla'$  is a widening.

*Proof.* Since  $\nabla'$  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 2.4.  $A_0$  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

$$A_2 = \{y - z \leq 1, z - y \leq 1\}$$

The inequalities in  $x, y$  are not stable, hence they are dropped. The inequalities in  $x, z$  are stable, but they were redundant in  $A_1$ , and are not entailed by the other stable inequalities in  $\text{complete}(A_1 \nabla (A_1 \sqcup C_1))$ , hence are dropped along with the unstable inequalities. Also observe that  $\text{complete}(A_2) = A_2$ , and that this is fixpoint.

Is this entirely 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 chimes 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 representation using the 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. Conclusion

This paper has looked at widening for polyhedral constraints, with a focus on weakly relational domains and TVPI in particular. It has illustrated the importance of capturing relations between variables whilst performing program analysis for verification. The key role of widening in this analysis has been highlighted. The application of widening is not always straightforward and some difficult cases highlighting this have been presented. The interaction between widening and the closed forms maintained in weakly relational domains has been investigated and a new widening has been introduced that gives a new way to sidestep the problem. The paper also contains a lot of questions, some of which require theoretical development of widening, others of which require empirical investigation of the performance of widening. Future work is to address these questions more fully by carrying out experimental work with real-life programs to illuminate where problematic structures occur and where the performance of widening needs to be improved.

**Acknowledgements** The authors would like to thank Marius Zicius for discussions on two variable per inequality constraints.

## References

- [1] P. Cousot, R. Cousot, 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, ACM Press, 1977, pp. 238–252.
- [2] P. Cousot, Principles of Abstract Interpretation, MIT Press, 2021.
- [3] P. Cousot, N. Halbwachs, Automatic Discovery of Linear Restraints among Variables of a Program, in: Principles of Programming Languages, ACM Press, 1978, pp. 84–97.
- [4] A. Simon, A. King, J. M. Howe, Two Variables per Linear Inequality as an Abstract Domain, in: Logic Based Program Development and Transformation, volume 2664 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 71–89.
- [5] A. Miné, The Octagon Abstract Domain, *Higher-Order and Symbolic Computation* 19 (2006) 31–100.
- [6] J. M. Howe, A. King, A. Simon, Incremental Closure for Systems of Two Variables Per Inequality, *Theoretical Computer Science* 768 (2019) 1–42.
- [7] J. M. Howe, A. King, Logahedra: a New Weakly Relational Domain, in: International Symposium on Automated Technology for Verification and Analysis, volume 5799 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 306–320.
- [8] T. Reinbacher, M. Függer, J. Brauer, Runtime verification of embedded real-time systems, *Formal Methods in System Design* 44 (2014) 203–239.
- [9] P. Cousot, R. Cousot, J. Feret, A. Miné, X. Rival, Why does ASTRÉE scale up?, *Formal Methods in System Design* 35 (2009) 229–264.
- [10] A. Miné, A New Numerical Abstract Domain Based on Difference-Bound Matrices, in:



- Second Symposium on Programs as Data Objects, volume 2053 of *Lecture Notes in Computer Science*, Springer, 2001, pp. 155–172.
- [11] G. Gange, Z. Ma, J. A. Navas, P. Schachte, H. Søndergaard, P. Stuckey, A Fresh Look at Zones and Octagons, *ACM Transactions on Programming Languages and Systems* 43 (2021) 1–51.
  - [12] P. Cousot, R. Cousot, 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, volume 631 of *Lecture Notes in Computer Science*, Springer, 1992, pp. 269–295.
  - [13] F. Bourdoncle, Efficient chaotic iteration strategies with widenings, in: Proceedings of the International Conference on Formal Methods in Programming and Their Applications, volume 735 of *Lecture Notes in Computer Science*, Springer, 1993, pp. 128–141.
  - [14] R. Bagnara, P. M. Hill, E. Mazzi, E. Zaffanella, Precise widening operators for convex polyhedra, *Science of Computer Programming* 58 (2005) 28–56.
  - [15] B. Blanchet, P. Cousot, R. Cousot, J. Feret, Mauborgne, A. Miné, D. Monniaux, X. Rival, 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*, volume 2566 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 85–108.
  - [16] A. Simon, A. King, Widening Polyhedra with Landmarks, in: Proceedings of the 4th Asian Symposium on Programming Languages and Systems, volume 4279 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 166–182.
  - [17] A. Simon, *Value-Range Analysis of C Programs*, Springer, 2008.
  - [18] D. Gopan, T. W. Reps, Lookahead Widening, in: Proceedings of the 18th International Conference on Computer Aided Verification, volume 4144 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 452–466.
  - [19] A. Simon, L. Chen, Simple and precise widenings for polyhedra, in: Proceedings of the 8th Asian Symposium on Programming Languages and Systems, volume 6461 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 139–155.
  - [20] L. Gonnord, N. Halbwegs, Combining Widening and Acceleration in Linear Relation Analysis, in: Proceedings of the 13th International Static Analysis Symposium, volume 4134 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 144–160.
  - [21] M. Brain, S. Joshi, D. Kroening, P. Schrammel, Safety Verification and Refutation by k-Invariants and k-Induction, in: Proceedings of the 22th International Static Analysis Symposium, volume 9291 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 145–161.
  - [22] G. Gange, A. Navas, J. P. Schachte, H. Søndergaard, P. Stuckey, Dissecting widening: Separating termination from information, in: Proceedings of the 17th Asian Symposium on Programming Languages and Systems, volume 11893 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 95–114.
  - [23] R. Bagnara, P. M. Hill, E. Mazzi, E. Zaffanella, Widening operators for weakly-relational numeric abstractions, in: Proceedings of the 12th International Symposium on Static Analysis, volume 3672 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 3–18.
  - [24] R. Bagnara, P. M. Hill, E. Zaffanella, Weakly-relational Shapes for Numeric Abstractions: Improved Algorithms and Proofs of Correctness, *Formal Methods in System Design* 35

(2009) 279–323.

- [25] J. Feret, Static Analysis of Digital Filters, in: Proceedings of the 13th European Symposium on Programming, volume 2986 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 33–48.