Sum of abstract domains

Gianluca Amato, Simone Di Nardo Di Maio, and Francesca Scozzari

Università di Chieti-Pescara – Italy
{gamato,simone.dinardo,fscozzari}@unich.it

Abstract. In the abstract interpretation theory, program properties are encoded by abstract domains, and the combination of abstract domains leads to new properties to be analyzed. We propose a new method to combine numerical abstract domains based on the Minkowski sum. We provide a general framework equipped with all the necessary abstract operators for static analysis of imperative languages.

1 Introduction

The theory of abstract interpretation [8, 9] is based on the notion of abstract domain. The choice of the abstract domain determines the properties to be analyzed, the precision of the analysis and, in most cases, its computational complexity. In the literature on abstract interpretation, we find a large number of numerical abstract domains, such as intervals [7], polyhedra [11], octagons [15], zonotopes [13], parallelotopes [3] and polyhedra template [16]. The choice of an abstract domain is mainly guided by a trade off between analysis precision and complexity.

Abstract domains can also be combined or refined to obtain new abstract domains. The very first and fundamental method to combine two abstract domains is Cousot and Cousot reduced product [9]. Other methods include powerset [9], quotient [6], open products [5] and donut domains [12]. In many cases domain combinators cannot be applied blindly, but the resulting domain needs some tweaking, such as the design of specific abstract operators or an ad-hoc representation for abstract objects.

In this paper we introduce a new domain combinator based on the *Minkowski sum*. Given two sets $\mathcal{A}, \mathcal{B} \subseteq \mathbb{R}^n$, the (Minkowski) sum of $\mathcal{A}$ and $\mathcal{B}$ is the subset of $\mathbb{R}^n$ given by

$$\mathcal{A} + \mathcal{B} = \{\boldsymbol{a} + \boldsymbol{b} \in \mathbb{R}^n \mid \boldsymbol{a} \in \mathcal{A}, \boldsymbol{b} \in \mathcal{B}\} \enspace ,$$

where $\boldsymbol{a} + \boldsymbol{b}$ is the vector addition of the points $\boldsymbol{a}$ and $\boldsymbol{b}$. In other words, the Minkowski sum is the union of all the translations of the points in $\mathcal{A}$ by a point in $\mathcal{B}$. For instance, given the segments

$$\mathcal{A} = \{(x, 0) \in \mathbb{R}^2 \mid 0 \leq x \leq 1\}$$
$$\mathcal{B} = \{(0, y) \in \mathbb{R}^2 \mid 0 \leq y \leq 1\}$$

the Minkowski sum $\mathcal{A} + \mathcal{B}$ is the unit square $\mathcal{C} = \{(x, y) \in \mathbb{R}^2 \mid 0 \leq x \leq 1, 0 \leq y \leq 1\}$.

In our proposal, given any two numerical abstract domains A and B, we define a new abstract domain $\mathsf{A}+\mathsf{B}$ whose abstract objects are defined as the sum of an object in A and an object in B.

The Minkowski sum is well-suited to define a domain combinator, since it enjoys many geometric and algebraic properties (commutes with convex hull, distributes over the scalar product, admits an identity element and an annihilator) which greatly help in defining the abstract operators in the sum domain. Moreover, sum is not idempotent, so that, for an abstract domain A, in the general case we have that $\mathsf{A} \neq \mathsf{A}+\mathsf{A}$. This allows the construction of a new domain even from a single abstract domain. In this way, the sum combinator may be used as a domain refinement operator.

Minkowski sum has also been recently used to define the numerical abstract domain of zonotopes, which are bounded polyhedra generated as the sum of a finite number of segments. In some way, the sum domain combinator may be thought of as the lifting of the zonotope construction to the level of abstract domains.

In the rest of the paper we describe the theoretical foundation of the sum of abstract domains. Its abstract operators are designed by exploiting the operators of the original abstract domains, thus ensuring ease of implementation. A prototype has been developed for the Jandom static analyzer [2, 1, 4]. We show some experiments for the special case of the sum of the interval and parallelotope domains, and discuss some heuristics which may be used to enhance precision.

2 Notations

2.1 Linear Algebra

We denote by $\overline{\mathbb{R}}$ the set of real numbers extended with $+\infty$ and $-\infty$. Addition and multiplication are extended to $\overline{\mathbb{R}}$ in the obvious way. We use boldface for elements $\boldsymbol{v}$ of $\overline{\mathbb{R}}^n$. Any vector $\boldsymbol{v} \in \overline{\mathbb{R}}^n$ is intended as a column vector, and $\boldsymbol{v}^T$ is the corresponding row vector. Given $\boldsymbol{u}, \boldsymbol{v} \in \overline{\mathbb{R}}^n$, and a relation $\bowtie \in \{<, >, \leq, \geq, =\}$, we write $\boldsymbol{u} \bowtie \boldsymbol{v}$ if and only if $u_i \bowtie v_i$ for each $i \in \{1, \ldots, n\}$. We denote by $\inf_{\boldsymbol{u} \in \mathcal{A}} f(\boldsymbol{u})$ the greatest lower bound in $\overline{\mathbb{R}}$ of the set $\{f(\boldsymbol{u}) \mid \boldsymbol{u} \in \mathcal{A}\}$ and by $\mathbb{R}(m, n)$ the set of real matrices with m rows and n columns.

2.2 Abstract interpretation

In this paper we adopt a framework for abstract interpretation which is weaker than the common one based on Galois' connections/insertions (see [10, Section 7]). Given a poset $(\mathsf{C}, \leq^{\mathsf{C}})$ — the *concrete domain* — and a set A — the *abstract domain* — we establish an abstract–concrete relationship between them with the use of a *concretization map*, which is just a function $\gamma : \mathsf{A} \to \mathsf{C}$.

We say that $a \in \mathsf{A}$ is a *correct abstraction* of $c \in \mathsf{C}$ when $c \leq^{\mathsf{C}} \gamma(a)$. In general, a given $c \in \mathsf{C}$ has many correct abstractions. We say that $a \in \mathsf{A}$ is a *minimal correct abstraction* of $c \in \mathsf{C}$ when a is a correct abstraction of c and there is no $a' \in \mathsf{A}$ such that $c \leq^{\mathsf{C}} \gamma(a') <^{\mathsf{C}} \gamma(a)$. Moreover, $a \in \mathsf{A}$ is an *optimal* abstraction of $c \in \mathsf{C}$ when $c \leq^{\mathsf{C}} \gamma(a')$ implies $\gamma(a) \leq^{\mathsf{C}} \gamma(a')$.

A function $f^{\mathsf{A}} : \mathsf{A} \to \mathsf{A}$ is a correct abstraction of $f : \mathsf{C} \to \mathsf{C}$ when it preserves correctness of abstractions, i.e. when $c \leq^{\mathsf{C}} \gamma(a)$ implies $f(c) \leq^{\mathsf{C}} \gamma(f^{\mathsf{A}}(a))$. It is a minimal correct abstraction of $f : \mathsf{C} \to \mathsf{C}$ when it is correct and, for any $a \in \mathsf{A}$, $f^{\mathsf{A}}(a)$ is a minimal correct approximation of $f(\gamma(a))$. Analogously we define the concept of optimal abstraction for f. Composition preserves correctness, but not minimality and optimality. The best precision is reached when f^{A} is γ-complete, i.e., when $\gamma(f^{A}(a)) = f(\gamma(a))$.

An *abstraction function* is a map $\alpha : \mathsf{C} \to \mathsf{A}$ such that $c \leq^{\mathsf{C}} \gamma(\alpha(c))$. When an abstraction function exists (which is quite common), a correct abstraction of $f : \mathsf{C} \to \mathsf{C}$ may be defined as $\alpha \circ f \circ \gamma$. An abstraction function is minimal when, for each $c \in \mathsf{C}$, $\alpha(c)$ is a minimal correct abstraction of c. In this case, $\alpha \circ f \circ \gamma$ is a minimal correct abstraction of f. Analogously we define the concept of optimal abstraction function.

The abstract–concrete relationship induces a pre-order $\leq^{\mathsf{A}}$ on A defined as $a^1 \leq^{\mathsf{A}} a^2$ iff $\gamma(a^1) \leq^{\mathsf{C}} \gamma(a^2)$. Note that γ is a monotone map from $(\mathsf{A}, \leq^{\mathsf{A}})$ to $(\mathsf{C}, \leq^{\mathsf{C}})$. When $\leq^{\mathsf{A}}$ is a partial order and α is optimal, we have the classical framework based on Galois's insertions. A *widening* on A is a map $\nabla : \mathsf{A} \times \mathsf{A} \to A$ such that $a, a' \leq^{\mathsf{A}} a \nabla a'$ and for every sequence $x^0, \ldots, x^i, \ldots$ in A, the sequence $y^0 = x^0 \leq^{\mathsf{A}} \cdots \leq^{\mathsf{A}} y^{i+1} = y^i \nabla x^{i+1} \leq^{\mathsf{A}} \cdots$ is not strictly increasing.

2.3 Numerical domains

In the following, we recall the definition of several standard *numerical abstract domains*, i.e, abstractions of the concrete domain $(\wp(\mathbb{R}^n), \subseteq)$. We consider fixed the dimension n of the concrete domain. A set $\mathcal{A} \subseteq \mathbb{R}^n$ is called a *closed box* when there are $\boldsymbol{l}, \boldsymbol{u} \in \overline{\mathbb{R}}^n$ such that $\mathcal{A} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$, a *parallelotope* when there are an $n \times n$ invertible matrix A and $\boldsymbol{l}, \boldsymbol{u} \in \overline{\mathbb{R}}^n$ such that $\mathcal{A} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{l} \leq A\boldsymbol{x} \leq \boldsymbol{u}\}$, a *zonotope* when there is $A \in \mathbb{R}(m, n)$ such that $\mathcal{A} = \left\{A\binom{1}{\boldsymbol{\epsilon}} \mid \boldsymbol{\epsilon} \in [-1, 1]^{m-1}\right\}$ and a *polyhedral set* when there is $A \in \mathbb{R}(m, n)$ and $\boldsymbol{b} \in \overline{\mathbb{R}}^m$ such that $\mathcal{A} = \{\boldsymbol{x} \in \mathbb{R}^n \mid A\boldsymbol{x} \leq \boldsymbol{b}\}$.

The abstract objects of the interval, parallelotope, zonotope and polyhedral domains are, respectively, closed boxes, parallelotopes, zonotopes and polyhedral sets. Abstract objects are ordered by set inclusion, and the concretization map is the identity. In actual implementations, a finite representation is used for these abstract objects, but this is not relevant to our paper.

3 Combining domains by Minkowski sum

One of the most important operations in geometry, in particular in convexity theory, is the *Minkowski sum* of two sets.

Definition 1 (Minkowski sum). *Given two sets $A, B \subseteq \mathbb{R}^n$, the* Minkowski sum $\mathcal{A} + \mathcal{B} \subseteq \mathbb{R}^n$ *is defined as:*

$$\mathcal{A} + \mathcal{B} = \{\boldsymbol{a} + \boldsymbol{b} \in \mathbb{R}^n \mid \boldsymbol{a} \in \mathcal{A}, \boldsymbol{b} \in \mathcal{B}\} \ .$$

It is immediate to see that every element of the interval domain is the Minkowski sum of (possibly unbounded) segments. Moreover, any zonotope is the Minkowski sum of a finite number of bounded segments.

We introduce a new operator for combining two numerical abstract domains into a new domain whose objects are the sum of the abstract objects of the constituent domains.

Definition 2 (Sum of abstract domains). *Given two numerical abstract domains* A *and* B*, we define a new abstract domain called the* (Minkowski) sum *of* A *and* B*, which is:*

$$\mathsf{A} + \mathsf{B} = \{\langle A + B \rangle \mid A \in \mathsf{A}, B \in \mathsf{B}\}$$

with concretization map:

$$\gamma^{A+B}(\langle A + B \rangle) = \gamma^{\mathsf{A}}(A) + \gamma^{\mathsf{B}}(B) \ .$$

We use the notation $\langle A + B \rangle$ instead of (A, B), since the former better conveys the real purpose of the pair. We stress out that $\langle A + B \rangle$ is only a formal sum.

Example 3. Let $\mathcal{A} \in \mathsf{Int}$ and $\mathcal{B} \in \mathsf{Parallelotope}$ with

$$\begin{aligned} \mathcal{A} &= \{0 \leq x \leq 1, 0 \leq y \leq \infty\} \\ \mathcal{B} &= \{0 \leq y \leq 2, 0 \leq x - y \leq 2\} \end{aligned}$$

as depicted in Fig. 1(a) and 1(b). Then, $\langle \mathcal{A} + \mathcal{B} \rangle$ is an abstract object in $\mathsf{Int} + \mathsf{Parallelotope}$ such that

$$\gamma(\langle \mathcal{A} + \mathcal{B} \rangle) = \{0 \leq x \leq 3, 0 \leq y \leq \infty, x - y \leq 3\}$$

as depicted in Fig. 1(c). It is neither an interval nor a parallelotope nor a zonotope (since it is unbounded and has constraints on three different linear forms).

3.1 Ordering

The subset ordering $\subseteq$ on the concrete domain induces a pre-order $\leq^{\mathsf{A}+\mathsf{B}}$ on $\mathsf{A} + \mathsf{B}$. This is not a partial order, since different objects in $\mathsf{A} + \mathsf{B}$ represent the same concrete object. For example, in $\mathsf{Int} + \mathsf{Int}$, the objects $\langle [0, 1] + [0, 1] \rangle$ and $\langle [0, 0] + [0, 2] \rangle$ both represent the interval $[0, 2] \subseteq \mathbb{R}$.

Moreover, given objects $\langle A+B \rangle$ and $\langle A'+B' \rangle$, deciding whether $\langle A+B \rangle \leq^{\mathsf{A}+\mathsf{B}} \langle A' + B' \rangle$ is not an easy task. There are some sufficient conditions which ensure the required property, such as $A \leq^{\mathsf{A}} A'$ and $B \leq^{\mathsf{B}} B'$. When A and B are both abstractions of a domain C (often C is the polyhedra domain), then we may compute, on the domain C, the representation of A, B, A', B', the Minkowski sums $A + B$ and $A' + B'$, and check if the ordering holds. However, in the general case, an algorithm for deciding $\leq^{\mathsf{A}+\mathsf{B}}$ must be especially designed for a given instance of the sum combinator. In any case, we will show later that this is not required for the analysis.

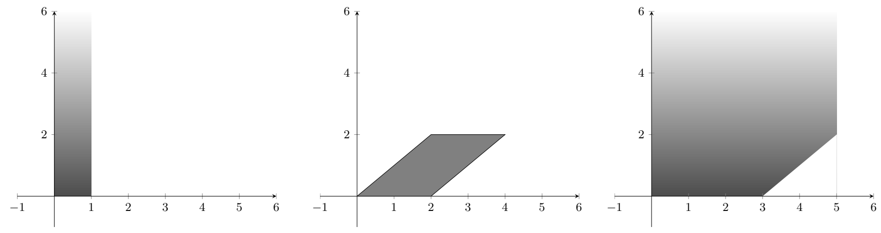

(a) Unbounded Box (b) Parallelotope (c) Minkowski Sum

Fig. 1. Minkowski sum of a box and a parallelotope.

3.2 Sum of standard domains

The following proposition summarizes some basic results when combining intervals, zonotopes, parallelotopes and polyhedra. It is worth noting that, in general, for an abstract domain A we have that $\mathsf{A} + \mathsf{A} \neq \mathsf{A}$. This is the case for the abstract domain of parallelotopes, since the sum of two parallelotopes is not, in general, a parallelotope, as shown in Figure 1. Moreover, given two domains A and B such that A is an abstraction of B, it may well happen that the sum of A and B is more concrete than both domains, as shown in the next theorem.

Theorem 4. *The abstract domains* Int*,* $\mathsf{Zonotope}$ *and* $\mathsf{Polyhedra}$ *are closed by Minkowski sum, that is:*

- $\mathsf{Int} + \mathsf{Int} = \mathsf{Int}$
- $\mathsf{Zonotope} + \mathsf{Zonotope} = \mathsf{Zonotope}$
- $\mathsf{Polyhedra} + \mathsf{Polyhedra} = \mathsf{Polyhedra}$

Moreover, the following inclusions are strict:

- $\mathsf{Zonotope} \subsetneq \mathsf{Int} + \mathsf{Zonotope}$
- $\mathsf{Parallelotope} \subsetneq \mathsf{Int} + \mathsf{Parallelotope}$
- $\mathsf{Parallelotope} \subsetneq \mathsf{Parallelotope} + \mathsf{Parallelotope}$

Figure 1 shows the counterexamples for the second part of the theorem. Note that, the box in Figure 1(a) is also a parallelotope, and the parallelotope in Figure 1(b) is also a zonotope, while their sum fails to be a zonotope.

4 Abstract operators

We now consider the operations on $\wp(\mathbb{R}^n)$ commonly used when defining the collecting semantics of imperative programming languages, and for each of them we introduce a correct approximation. We show that some abstract operators are γ-complete, provided the corresponding abstract operators on the component domain are also γ-complete.

In the following we fix two numerical abstract domains A and B and their sum $\mathsf{A} + \mathsf{B}$.

4.1 Union

Abstract union on the sum domain can be defined component-wise from the abstract unions of the two original domains.

Definition 5 (Abstract union). *Given $A_1, A_2 \in \mathsf{A}$ and $B_1, B_2 \in \mathsf{B}$, we define the abstract union $\cup^{\mathsf{A}+\mathsf{B}}$ as:*

$$\langle A_1 + B_1 \rangle \cup^{\mathsf{A}+\mathsf{B}} \langle A_2 + B_2 \rangle = \langle (A_1 \cup^{\mathsf{A}} A_2) + (B_1 \cup^{\mathsf{B}} B_2) \rangle \ .$$

Theorem 6. *The abstract union is correct.*

4.2 Linear transformations

A linear (homogeneous) assignment has the form $x_i := \boldsymbol{a}^T \boldsymbol{x}$ where $\boldsymbol{a} \in \mathbb{R}^n$ and $\boldsymbol{x}$ is the vector of program variables. Linear assignments (even multiple linear assignments) may be represented as linear transformations in $\mathbb{R}^n$. If M is a square real matrix of order n and $\mathcal{A} \subseteq \mathbb{R}^n$, we consider the operator

$$M \cdot \mathcal{A} = \{ M\boldsymbol{a} \mid \boldsymbol{a} \in \mathcal{A} \} \ .$$

The abstraction of $\cdot$ in $\mathsf{A} + \mathsf{B}$ may be easily recovered by its abstraction on A and B.

Definition 7 (Linear assignment). *Given $A \in \mathsf{A}$, $B \in \mathsf{B}$ and $M \in \mathbb{R}(n, n)$ we define the abstract linear transformation as:*

$$M \cdot^{\mathsf{A}+\mathsf{B}} \langle A + B \rangle = \langle M \cdot^{\mathsf{A}} A + M \cdot^{\mathsf{B}} B \rangle \ .$$

Theorem 8. *The abstract linear assignment operator is correct. Moreover, it is γ-complete if the corresponding abstract operators on A and B are γ-complete.*

4.3 Translations

Given $\boldsymbol{b} \in \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^n$, consider the translation operator

$$\mathcal{A} + \boldsymbol{b} = \mathcal{A} + \{\boldsymbol{b}\} = \{ \boldsymbol{a} + \boldsymbol{b} \mid \boldsymbol{a} \in \mathcal{A} \} \ .$$

As for linear transformations, it is easy to determine a correct abstraction of $+$ in the abstract domain $\mathsf{A} + \mathsf{B}$ starting from correct abstractions in A and B, but there is not a single abstract version which could be considered the canonical one.

Definition 9 (Abstract translation). *Given $A \in \mathsf{A}$, $B \in \mathsf{B}$, $\boldsymbol{b} \in \mathbb{R}^n$ and $w \in \mathbb{R}$, we define the abstract sum (weighted by w) as*

$$\langle A + B \rangle +_w^{\mathsf{A}+\mathsf{B}} \boldsymbol{b} = \langle (A +^{\mathsf{A}} w\boldsymbol{b}) + (B +^{\beta} (1 - w)\boldsymbol{b}) \rangle \ .$$

In this definition, the weight w determines in which part of the two abstract objects A and B we need to apply the translation. It may be applied entirely on A ($w = 1$), entirely in B ($w = 0$) or divided between them.

Theorem 10. *The abstract translation operator is correct. Moreover, it is γ-complete if the corresponding abstract operators on A and B are γ-complete.*

4.4 Non-deterministic assignment

Given $i \in \{1, \dots, n\}$, we define the concrete operator $\mathsf{forget}_i : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ as

$$\mathsf{forget}_i(\mathcal{A}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{a} \in \mathcal{A} \wedge \forall j \neq i.\ x_i = a_i\} \ .$$

This simulates the effect of a non-deterministic assignment $x_i :=?$.

Definition 11. *Given $i \in \{1, \dots, n\}$, we define the non-deterministic assignment as*

$$\mathsf{forget}_i^{\mathsf{A+B}}(\langle A + B \rangle) = \langle \mathsf{forget}_i^{\mathsf{A}}(A) + B \rangle$$

or

$$\mathsf{forget}_i^{\mathsf{A+B}}(\langle A + B \rangle) = \langle A + \mathsf{forget}_i^{\mathsf{B}}(B) \rangle \ .$$

Both definitions are correct, and the choice between them follows by heuristic considerations. We will talk about this later in the paper.

Theorem 12. *The abstract non-deterministic assignment is correct. Moreover, it is γ-complete if the corresponding abstract operator on* A *(for the first form) or* B *(for the second form) is γ-complete.*

4.5 Refinement by linear inequality

The concrete refinement by linear inequality $\mathsf{refine}_{(\boldsymbol{a},b)}$, with $\boldsymbol{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, is the intersection of a subset of $\mathbb{R}^n$ with an half-space. Formally:

$$\mathsf{refine}_{(\boldsymbol{a},b)}(\mathcal{A}) = \mathcal{A} \cap \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{a}^T \boldsymbol{x} \leq b\} \ .$$

In the following, we extend this definition to the case $b = \pm\infty$ with the obvious interpretation.

Definition 13. *Given $\boldsymbol{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, we define the abstract refinement by linear inequality as*

$$\mathsf{refine}_{(\boldsymbol{a},b)}^{\mathsf{A+B}}(\langle A + B \rangle) = \langle \mathsf{refine}_{(\boldsymbol{a},b-d_2)}^{\mathsf{A}}(A) + \mathsf{refine}_{(\boldsymbol{a},b-d_1)}^{\mathsf{B}}(B) \rangle$$

where $d_1, d_2 \in \overline{\mathbb{R}}$ such that $d_1 \leq \inf_{\boldsymbol{x} \in \mathcal{A}} \boldsymbol{a}^T \boldsymbol{x}$ and $d_2 \leq \inf_{\boldsymbol{x} \in B} \boldsymbol{a}^T \boldsymbol{x}$. Moreover, we define

$$\mathsf{refine}_{(\boldsymbol{a},+\infty)}^{\mathsf{A+B}}(\langle A + B \rangle) = (\langle A + B \rangle)$$
$$\mathsf{refine}_{(\boldsymbol{a},-\infty)}^{\mathsf{A+B}}(\langle A + B \rangle) = C$$

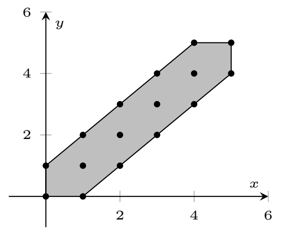
where C is any correct approximation of $\emptyset$ (i.e., any value in $\mathsf{A} + \mathsf{B}$*).*

This operator needs a way to determine a lower bound for the value that a linear form may assume in every abstract object of the domains A and B. If this is not possible, both d_1 and d_2 may be considered to be $-\infty$, and the refine operator turns out to be the identity.

Theorem 14. *The operator $\mathsf{refine}_{(\boldsymbol{a},b)}^{\mathsf{A+B}}$ is correct.*

```
i = 0
x = 0
y = 0
while (i <= 4) {
  i = i+1
  if (?) x = i−1 else x = i
  if (?) y = i−1 else y = i
}
```

(a) Example program.



(b) Possible values for variables x and y at the last program point of the loop.

Fig. 2. Example program.

4.6 Widening and Narrowing

Given $A_1, A_2 \in \mathsf{A}$ and $B_1, B_2 \in \mathsf{B}$ we can use the widening/narrowing operators of the individual numerical abstract domains to devise widening/narrowing operators for the Minkowski sum.

Definition 15. *The abstract widening for* $\mathsf{A}+\mathsf{B}$ *is defined as*

$$\langle A_1 + B_1 \rangle \nabla^{\mathsf{A}+\mathsf{B}} \langle A_2 + B_2 \rangle = \langle A_1 \nabla^{\mathsf{A}} A_2 + B_1 \nabla^{\mathsf{B}} B_2 \rangle$$

and the abstract narrowing for $\mathsf{A}+\mathsf{B}$ *is defined as*

$$\langle A_1 + B_1 \rangle \Delta^{\mathsf{A}+\mathsf{B}} \langle A_2 + B_2 \rangle = \langle A_1 \Delta^{\mathsf{A}} A_2 + B_1 \Delta^{\mathsf{B}} B_2 \rangle \ .$$

Theorem 16. *The abstract operator* $\nabla^{\mathsf{A}+\mathsf{B}}$ *is a widening and* $\Delta^{\mathsf{A}+\mathsf{B}}$ *is a narrowing.*

Note that widening is defined component-wise. This means that, at widening points, the increasing chains we get are of the form

$$\langle A_0 + B_0 \rangle \leq^{\mathsf{A}+\mathsf{B}} \langle A_1 + B_1 \rangle \leq^{\mathsf{A}+\mathsf{B}} \langle A_2 + B_2 \rangle \leq^{\mathsf{A}+\mathsf{B}} \dots$$

with $A_0 \leq^{\mathsf{A}} A_1 \leq^{\mathsf{A}} A_2 \dots$ and $B_0 \leq^{\mathsf{A}} B_1 \leq^{\mathsf{A}} B_2 \dots$. Both the chains of A_i's and B_i's eventually stop increasing, since they are constrained by ∇^{A} and ∇^{B}. Therefore, if we can decide $\leq^{\mathsf{A}}$ and $\leq^{\mathsf{B}}$, then we known when to stop the analysis even if, in the general case, we cannot decide $\leq^{\mathsf{A}+\mathsf{B}}$.

5 An example

Consider the program in Figure 2(a) and the graph in Figure 2(b) which depicts the possible values for variables x and y at the end of the loop's body. The convex hull of these points, which is the shaded area in the figure, may be described

```
i = x = y = 0
[$i = 0, x = 0, y = 0$] + [$i = 0, x = 0, y = 0$]
while (i <= 4) {
  [$i = 0, x = 0, y = 0$] + [$i = 0, x = 0, y = 0$]
  i = i+1
  [$i = 0, x = 0, y = 0$] + [$i = 1, x = 0, y = 0$]
  if (?)
    x = i−1
    [$i = 0, x = -1, y = 0$] + [$i = 1, x = i, y = 0$]
  else
    x = i
    [$i = 0, x = 0, y = 0$] + [$i = 1, x = i, y = 0$]
  [$i = 0, -1 \leq x \leq 0, y = 0$] + [$i = 1, x = i, y = 0$]
  if (?) y = i−1 else y = i
  [$i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0$] + [$i = 1, x = i, y = i$]
}
```

Fig. 3. Annotated program after the 1st loop iteration.

as the sum of the box $\{i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0\}$ and the parallelotope $\{1 \leq i \leq 5, x = i, y = i\}$. Using the domain Int + Parallelotope, we are able to infer this property, although we need to refine the raw domain operators with heuristics specifically tailored for this specific combination.

The main tunable aspect of the operators we have described in the previous section is the value of the weight w in translations and abstractions. Choosing w randomly may lead to very bad precision. For the moment, assume we choose $w = 0$ for the increment $i = i + 1$ and $w = 1$ for all the other assignments. Figure 3 shows the candidate invariants reached after a single iteration of the while loop. Using the standard domain operators, before entering the while loop we get the invariant $\{i = 0, x = 0, y = 0\} + \{i = 0, x = 0, y = 0\}$ which is preserved by the loop's guard. The increment to i, according to the chosen value $w = 0$, yields $\{i = 0, x = 0, y = 0\} + \{i = 1, x = 0, y = 0\}$.

The true branch of the first non-deterministic conditional statement leads to $\{i = 0, x = -1, y = 0\} + \{i = 1, x = i, y = 0\}$ while the else branch leads to $\{i = 0, x = 0, y = 0\} + \{i = 1, x = i, y = 0\}$. The component-wise union gives $\{i = 0, -1 \leq x \leq 0, y = 0\} + \{i = 1, x = i, y = 0\}$. Repeating the same argument for the second conditional statement we get $\{i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0\} + \{i = 1, x = i, y = i\}$.

At the second iteration, the previous while invariant $\{i = 0, x = 0, y = 0\} + \{i = 0, x = 0, y = 0\}$ is widened with $\{i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0\} + \{i = 1, x = i, y = i\}$ to get $\{i = 0, -\infty < x \leq 0, -\infty < y \leq 0\} + \{0 \leq i < \infty, x = i, y = i\}$. This is the fix-point of the ascending chain. The subsequent descending phase yields the while invariant $\{i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0\} + \{0 \leq i \leq 5, x = i, y = i\}$. The program with the final annotations is shown

```
i = x = y = 0
$[i = 0, x = 0, y = 0] + [i = 0, x = 0, y = 0]$
while (i <= 4) {
  $[i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0] + [0 \leq i \leq 4, x = i, y = i]$
  i = i+1
  $[i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0]+[1 \leq i \leq 5, x = i-1, y = i-1]$
   if (?)
    x = i-1
    $[i = 0, x = -1, -1 \leq y \leq 0] + [1 \leq i \leq 5, x = i, y = i - 1]$
  else
    x = i
    $[i = 0, x = 0, -1 \leq y \leq 0] + [1 \leq i \leq 5, x = i, y = i - 1]$
  $[i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0] + [1 \leq i \leq 5, x = i, y = i - 1]$
  if (?) y = i-1 else y = i
  $[i = 0, -1 \leq x \leq 0, -1 \leq y \leq 0] + [1 \leq i \leq 5, x = i, y = i]$
}
```

Fig. 4. Annotated program at the end of the analysis. The highlighted invariant is the one sought after.

in Figure 4. The invariant at the last program point in the loop is the one we were looking for.

Crucial in obtaining the desired result is that the parallelotope component encodes the unsound relationship $i = x = y$ while the box component contains the deviation w.r.t. this line which makes the result correct. If we use $w = 1$ for $i = i + 1$ instead of $w = 0$, the initial invariant $i = x = y = 0$ for the parallelotope component remains stable for the entire while loop, and all the analysis actually proceeds on the interval domain. The result is the much less precise $\{1 \leq i \leq 5, 0 \leq x \leq 5, 0 \leq y \leq 5\} + \{i = 0, x = i, y = i\}$. On the contrary, if we use $w = 0$ for all the assignments, the analysis actually proceeds in the parallelotope domain. The result depends on the heuristics used for parallelotopes. The Jandom static analyzer determines the following: $\{i = 0, x = 0, y = 0\} + \{1 \leq i \leq 6, 1.0 \leq -i + x \leq 0.0, -1.0 \leq -i + y \leq 0.0\}$. The result is qualitatively better than the one on intervals, but not as good as the one with get with the correct choices for w.

The problem is how to determine an heuristic to choose the value of w for the assignment operators. Our idea is to use the parallelotope domain to capture an "ideal" relationships between variables, and resort to the interval domain for capturing the deviation w.r.t. the ideal behavior. This means that, for a non invertible assignment $x_i = \boldsymbol{a}^T\boldsymbol{x} + b$ we use $w = 1$, in the hope that $x_i - \boldsymbol{a}^T\boldsymbol{x}$ has an almost constant value in a program point, modulo some variability captured by the interval domain. For all other assignments we use $w = 0$. In the program of Figure 2(a) this heuristic yields the optimal choice we have shown before.

6 Precision of abstract operators

In this section we reason about the precision of the abstract operators of the sum domain. We will see that even very precise operators (such as translations) are problematic due to the fact that many different representation exists for the same abstract object, and that the imprecise operators gives different results for different representations. Finally, we explicitly discuss the domain $\mathsf{Int} + \mathsf{Parallelotope}$.

6.1 An approximate ordering

The subset ordering $\subseteq$ in $\wp(\mathbb{R}^n)$ induces the pre-order $\leq^{\mathsf{A}+\mathsf{B}}$ on $\mathsf{A}+\mathsf{B}$. However, many of the operators we have defined are not monotonic w.r.t. $\leq^{\mathsf{A}+\mathsf{B}}$. This makes difficult to reason about the precision of analysis. We do not even know if, improving the precision of one operator, actually improves the precision of the result. However, it is possible to define a coarser ordering on $\mathsf{A}+\mathsf{B}$, component-wise as

$$\begin{aligned}\langle A_1 + B_1\rangle \sqsubseteq^{\mathsf{A}+\mathsf{B}} \langle A_2 + B_2\rangle \iff{} & \exists \boldsymbol{p} \in \mathbb{R}^n \text{ s.t.} \\ & \gamma(A_1) \subseteq \gamma(A_2) + \boldsymbol{p} \text{ and } \gamma(B_1) \subseteq \gamma(B_2) - \boldsymbol{p} \ .\end{aligned}$$

It turns out that $\langle A_1 + B_1\rangle \sqsubseteq^{\mathsf{A}+\mathsf{B}} \langle A_2 + B_2\rangle$ implies $\langle A_1 + B_1\rangle \leq^{\mathsf{A}+\mathsf{B}} \langle A_2 + B_2\rangle$. Moreover, all operators in Section 4 (but widening and narrowing) are monotone w.r.t. $\sqsubseteq^{\mathsf{A}+\mathsf{B}}$. Therefore, if we replace an abstract operator with another one which is more precise w.r.t. the $\sqsubseteq^{\mathsf{A}+\mathsf{B}}$ ordering, we are sure we are not going to loose precision globally for the entire analysis, modulo the effect of the non-monotonic widening and narrowing operators.

6.2 Abstraction function

Particularly critical, in the general definition of the sum domain, is the fact that we do not have a good abstraction function. Actually, a family of correct abstraction functions may be defined easily as follows.

Theorem 17. *Given abstract domain A and B with abstraction functions α^{A} and α^{B}, consider a weight $w \in \mathbb{R}$. Then*

$$\alpha_w^{\mathsf{A}+\mathsf{B}}(\mathcal{C}) = \langle \alpha^{\mathsf{A}}(w\mathcal{C}) + \alpha^{\mathsf{B}}((1-w)\mathcal{C})\rangle$$

is a correct abstraction function for any $w \in \mathbb{R}$.

However, in the general case $\alpha_w(\mathcal{C})$ is not a minimal abstraction of $\mathcal{C}$ for any value of w. Even if domains A and B have good abstraction functions α^{A} and α^{B}, the abstraction function $\alpha^{\mathsf{A}+\mathsf{B}}$ may have, in general, a bad precision. For example, consider Figure 2(b) and let $\mathcal{C}$ be the set of points in the shaded area. Although $\mathcal{C}$ may be described as the sum of a box and a parallelotope, there is no choice of weight w such that $\alpha_w^{\mathsf{A+B}}(\mathcal{C})$ returns such as description. Actually, we have

$$\alpha_w^{\mathsf{A+B}}(\mathcal{C}) = \langle A + B \rangle$$

with

$$A = \{0 \le x \le 5w, 0 \le y \le 5w\}$$
$$B = \{-(1-w) \le x - y \le 1 - w, 0 \le x + y \le 10(1-w)\}$$

and $\gamma^{\mathsf{A+B}}(\alpha_w^{\mathsf{A+B}}(\mathcal{C})) \supsetneq \mathcal{C}$.

In abstraction interpretation, the abstraction function is often used to guide the definition of the abstract operators. If $f : \mathcal{C} \to \mathcal{C}$ is a concrete operator, $f^{\mathsf{A}} = \alpha^{\mathsf{A}} \circ f \circ \gamma^{\mathsf{A}}$ is a correct abstract operator. However, since $\alpha^{\mathsf{A+B}}$ may have such a bad precision, this approach is not applicable for the sum domain. While for most concrete operators we were nonetheless able to find good abstract counterparts, this is definitively not easy for linear refinement.

6.3 Linear refinement

Consider the sum domain $\mathsf{Int}+\mathsf{Int}$. The box $\mathcal{B} = [0,2]\times[0,2]$ may be described in $\mathsf{Int}+\mathsf{Int}$ as $S = \langle [0,1]\times[0,1]+[0,1]\times[0,1] \rangle$. Assume we want to refine S with the linear inequality $x_1 \le 1$, i.e., we want to compute $\mathsf{refine}_{(\boldsymbol{a},b)}(S)$ with $\boldsymbol{a} = (1,0)^T$ and $b = 1$. The result is the box $[0,1]\times[0,2]$ which may be represented optimally as, for example, $\langle [0,1/2]\times[0,1] + [0,1/2]\times[0,1] \rangle$.

However, applying the definition for abstract refinement in Section 4, we get a much coarser result. Let $\mathcal{A} = [0,1]\times[0,1]$. Note that $\inf_{\boldsymbol{x}\in\mathcal{A}} \boldsymbol{a}^T\boldsymbol{x} = 0 = \inf_{\boldsymbol{x}\in\mathcal{B}} \boldsymbol{a}^T\boldsymbol{x} = 0$. By choosing $d_1 = d_2 = 0$, we get $\mathsf{refine}^{\mathsf{Int}}_{(\boldsymbol{a},b-d_1)}(\mathcal{A}) = \mathsf{refine}^{\mathsf{Int}}_{(\boldsymbol{a},b-d_2)}(\mathcal{A}) = \mathcal{A}$. Hence $\mathsf{refine}^{\mathsf{Int+Int}}_{(\boldsymbol{a},b)}(\langle \mathcal{A}+\mathcal{A} \rangle) = \langle \mathcal{A}+\mathcal{A} \rangle$.

Here the problem is caused by the high redundancy in $\mathsf{Int}+\mathsf{Int}$. The constraint $x_1 \le 2$ in $\mathcal{B}$ is divided between the two's $x_1 \le 1$ in $\langle \mathcal{A}+\mathcal{A} \rangle$. The same may happen in $\mathsf{Int} + \mathsf{Parallelotope}$ when the parallelotope has some equations of the kind $x_i \le b$. Therefore, in this sum, the parallelotope component should be tweaked to avoid generating constraints parallel to the axis.

Another problem caused by linear refinement is with unbounded abstract object. Consider in $\mathsf{Int}+\mathsf{Parallelotope}$ the full $\mathbb{R}^n$, represented as $\langle \mathbb{R}^n + \mathbb{R}^n \rangle$. If we want to refine with $\boldsymbol{a}^T\boldsymbol{x} \le b$, for any $\boldsymbol{a}$ and b, we get $\inf_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{a}^T\boldsymbol{x} = -\infty$, hence $\mathsf{refine}_{(\boldsymbol{a},b)}(\langle \mathbb{R}^n+\mathbb{R}^n \rangle) = \langle \mathsf{refine}_{(\boldsymbol{a},+\infty)}(\mathbb{R}^n) + \mathsf{refine}_{(\boldsymbol{a},+\infty)}(\mathbb{R}^n) \rangle = \langle \mathbb{R}^n+\mathbb{R}^n \rangle$. Note that, on the contrary, if we represent $\mathbb{R}^n$ as $\langle \{0\} + \mathbb{R}^n \rangle$, then refinement works much better, essentially performing the refinement on the second component. Here $\{0\}$ may be replaced by any one-point element without affecting precision.

In some way, both problems are related and could be solved by some form of normalization which, before applying linear refinement $\mathsf{refine}_{(\boldsymbol{a},b)}$ to $\langle A + B \rangle$, transform $\langle A + B \rangle$ to $\langle A' + B' \rangle$ with the same concretization but minimizing the range of $\{\boldsymbol{a}^T\boldsymbol{x} \mid \boldsymbol{x} \in \gamma(A')\}$. There is no general method to perform such a normalization, which should be devised specifically for each instance of the sum domain.

6.4 Union and Widening

Abstract union may also be quite imprecise. Consider the abstract values $A_1 = \langle\{\mathbf{0}\}+\{\mathbf{0}\}\rangle$ and $A_2 = \langle\{\boldsymbol{a}\}+\{\mathbf{0}\}\rangle$ on the domain $\mathsf{Int}+\mathsf{Parallelotope}$, where $\boldsymbol{a}$ is any vector in $\mathbb{R}^n$. The concrete union $\gamma(A_1)\cup\gamma(A_2)$ is the two point set $\{\mathbf{0}, \boldsymbol{a}\}$. In the $\mathsf{Int}+\mathsf{Parallelotope}$ domain, its optimal representation is $\langle\{\mathbf{0}\}+\mathcal{L}\rangle$ where $\mathcal{L}$ is the segment from $\mathbf{0}$ to $\boldsymbol{a}$. However, the abstract union gives $A_1 \cup^{\mathsf{Int}+\mathsf{Int}} A_2 = \langle\mathcal{B}+\{\mathbf{0}\}\rangle$ where $\mathcal{B}$ is the box with corners $\mathbf{0}$ and $\boldsymbol{a}$. This is (except for the case $\boldsymbol{a} = \mathbf{0}$) much worse than the optimal result.

The problem arises from the fact that abstract union cannot "restructure" the representation to use the strong points of the component domains. We believe that designing a more precise union operator is a difficult challenge. Widening, being defined component-wise as union, has similar problems.

6.5 Other operators

The other operators are generally much more precise than linear refinement.

non deterministic assignment) Many domains have γ-complete non-deterministic assignments. When this happens, it is better to apply non deterministic assignment to this component. If the other component has a γ-complete assignment of the constant 0 to a variable, we may refine forget_i as follows:

$$\mathsf{forget}_i^{\mathsf{A}+\mathsf{B}}(\langle A+B\rangle) = \langle \mathsf{forget}_i^{\mathsf{A}}(A) + M \cdot^{\mathsf{B}} B\rangle$$

where M is the matrix whose effect is to assign 0 to the i-th variable. This is better since it smaller w.r.t. $\sqsubseteq^{\mathsf{A}+\mathsf{B}}$ than the one defined in Section 4.

linear assignments) This operator does not cause precision problems.

translations) Although this operator is quite precise, the choice of the weight w is crucial in obtaining good results. This is because of the imprecision of the abstract union operator. An example of this phenomenon has been shown in Section 5.

6.6 The domain $\mathsf{Int} + \mathsf{Parallelotope}$

The domain $\mathsf{Int} + \mathsf{Parallelotope}$ is one of the simplest non trivial domains which may be obtained with the sum combinator. Since translations are γ-complete both in Int and $\mathsf{Parallelotope}$, the same holds for translations on $\mathsf{Int}+\mathsf{Parallelotope}$. For the same reason, non-deterministic assignment is γ-complete.

As we said before, deciding whether $\langle B_1 + P_1\rangle \leq \langle B_2 + P_2\rangle$ may be easily implemented by representing sums as convex polyhedra, and checking set inclusion. On the contrary, improving the abstraction function is harder. Even if the abstraction is firstly computed over polyhedra, it remains the problem of representing a polyhedron in the most effective way as sum of a box and a parallelotope. Solving the abstraction problem could lead to the design of better operators for union and refinement, which also suffer from great imprecision.

We have implemented a prototype of the sum combinator in the static analyzer Jandom and did some preliminary test of the Int + Parallelotope domain on the ALICe benchmarks [14] (plus some additional test programs). The test-suite comprises a total of 105 models with 316 program points. We have compared the results on the sum with the results on the parallelotopes (the comparison with the interval domain gives very similar results). With respect to the parallelotope domain, the sum is more precise on 53 program points and less precise in 72 program points, while in 76 cases the results are incomparable. We believe that these preliminary tests are very promising:

– The regressions w.r.t. the component domains were expected and are mainly due to the fact that some operators on the sum, like union, introduce a loss of precision.
– In addition to the cases where sum is better, we have many cases with incomparable results. This shows that the sum combinator is able to improve at least one constraint in many program points. Combining the results on the sum domain with the (incomparable) results on the component domain, we obtain more precise results in more than 40% of the program points.
– No special code has been written for the Int + Parallelotope domain: all the operators are the generic ones of the sum combinator, and the only heuristic applied is choosing w in translations as we have done in Section 5. Nonetheless, the domain was able to produce new constraints. This is in contrast with other combinators, such as reduced product, where new results always need some specific code.

We still need to do more experiments and find better heuristics, but the fact that we have found many new constraints is encouraging.

6.7 Analysis kickoff and non-deterministic assignments

Some numerical abstract domains, such as template parallelotope, template polyhedra and zonotopes, need a special treatment in the starting phase of the analysis. For instance, consider the domain of template polyhedra with constraints $x + y$ and $x - y$, and assume that we start the analysis of a program whose first statement is the assignment $x = 0$. Since we cannot represent this information with the given template, we loose the information about the variable x. A similar problem arises with zonotopes, where we cannot represent an unbounded value for y. Such a situation can be easily managed using the sum combinator, by considering the sum with a simple abstract domain, like intervals, which is exploited in the starting phase of the analysis.

More generally, we can use the sum combinator to enrich a domain which may only represent bounded objects (such as the zonotope domain) summing to it a simple domain able to represent unbounded objects (such as the interval domain). The resulting domain (Int + Zonotope in our example) would be able to handle unbounded objects and non-deterministic assignment with greater precision.

7 Conclusion

We have described the theoretical foundation of the sum of abstract domains. We have defined generic abstract operators which can be easily implemented exploiting the corresponding operators on the original domains and we have discussed possible improvements.

For the sum of intervals and parallelotopes we have also discussed some heuristics to enhance the precision of the analysis and presented preliminary experimental results.

References

1. Gianluca Amato, Simone Di Nardo Di Maio, and Francesca Scozzari. Numerical static analysis with Soot. In *Proceedings of the ACM SIGPLAN International Workshop on State of the Art in Java Program analysis*, SOAP, 2013.
2. Gianluca Amato and Francesca Scozzari. Jandom [Software]. Available from `https://github.com/jandom-devel/Jandom`.
3. Gianluca Amato and Francesca Scozzari. The abstract domain of parallelotopes. In *NSAD 2012. Proceedings*, volume 287 of *ENTCS*, pages 17–28. 2012.
4. Gianluca Amato and Francesca Scozzari. Localizing widening and narrowing. In *SAS 2013, Proceedings*, volume 7935 of *LNCS*, pages 25–42, 2013.
5. Agostino Cortesi, Baudouin Le Charlier, and Pascal Van Hentenryck. Combinations of abstract domains for logic programming: Open product and generic pattern construction. *Science of Computer Programmming*, 38(1–3):27–71, 2000.
6. Agostino Cortesi, Gilberto Filé, and William W. Winsborough. The quotient of an abstract interpretation. *Theoretical Computer Science*, 202(1–2):163–192, 1998.
7. Patrick Cousot and Radhia Cousot. Static determination of dynamic properties of programs. In *Proc. 2nd Int'l Symposium on Programming*, pages 106–130, 1976.
8. Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL '77: Proceedings*, pages 238–252, 1977.
9. Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *POPL '79: Proceedings*, pages 269–282, 1979.
10. Patrick Cousot and Radhia Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–549, 1992.
11. Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL '78: Proceedings*, pages 84–97, 1978.
12. Khalil Ghorbal, Franjo Ivancic, Gogul Balakrishnan, Naoto Maeda, and Aarti Gupta. Donut domains: Efficient non-convex domains for abstract interpretation. In *VMCAI 2012. Proceedings*, volume 7148 of *LNCS*, pages 235–250, 2012.
13. Eric Goubault, Sylvie Putot, and Franck Védrine. Modular static analysis with zonotopes. In *SAS 2012. Proceedings*, volume 7460 of *LNCS*, pages 24–40, 2012.
14. Vivien Maisonneuve, Olivier Hermant, and François Irigoin. Alice: A framework to improve affine loop invariant computation. In *5th Workshop on INvariant Generation*, 2014.
15. Antoine Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, 2006.
16. Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Scalable analysis of linear systems using mathematical programming. In *VMCAI 2005. Proceedings*, volume 3385 of *LNCS*, pages 25–41. 2005.