Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 287 (2012) 65–76

www.elsevier.com/locate/entcs

An Accurate Join for Zonotopes, Preserving Affine Input/Output Relations

Eric Goubault, Tristan Le Gall and Sylvie Putot

CEA, LIST, Laboratory for the Modelling and Analysis of Interacting Systems, Point courrier 174, 91191 Gif-sur-Yvette, France.

Abstract

Zonotopes are a convenient abstract domain for the precise analysis of programs with numerical variables. Compared to the domain of convex polyhedra, it is less expensive and may easily handle non-linear assignments. However, the classical join operator of this abstract domain does not always preserve linear invariants, unlike the convex hull. We present a *global join* operator that preserves some affine relations. We end up by showing some experiments conducted on the constrained Taylor1+ domain of Apron.

Keywords: Abstract interpretation, zonotopes, affine relations

1 Introduction

Zonotopic methods have proved useful in a number of contexts in computer science, such as image processing [13], reachability analysis of hybrid systems [9,6] and static analysis by abstract interpretation as implemented in FLUCTUAT [5,12]. Most transfer functions (in particular, the interpretation of arithmetic expressions and assignments) are precise and fast in zonotopes. But set-theoretic functions, such as the meet and join operations, are difficult to characterize and compute, contrarily to most of other sub-polyhedric domains (zones [15], linear templates [18], even polyhedra [4]). Indeed, they are non canonical operations, and can only be over-approximated in the general case.

In [8], we proposed a meet operation for our zonotopic abstract domains. In this article, we propose an improvement of the time and space efficient upper bound operators that we introduced in [10,11,7]. The problem of these upper bound operators is that they forget a lot about the relations between values that program variables can take. They are only proved optimal (that is, giving minimal upper

[1] Email:firstname.lastname@cea.fr

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2012.09.007

bounds) in the case where only one variable differs in the joined states, which is of course fairly disappointing. This paper constructs a much better "global" upper bound, that is proved to be optimal in some situations, involving any number of program variables.

Contributions of the paper

We prove in Section 3 that minimal upper bounds e for constrained affine sets e_1 and e_2 necessarily satisfy the affine relations between program variables and central *noise symbols* (used to encode the inputs of the program) satisfied by both e_1 and e_2. The main contribution of the article is then the definition of a more precise *global* join operator, that uses the previous component-wise join of [10,11,7] only on part of the variables, and these common affine relations to deduce an upper bound on the other variables. We actually show in Proposition 3.3 that our join is optimal in some important situations.

In Section 4, we show on some applications that problems that would normally involve some form of disjunctive analysis or need some form of "clock domain" [3], can be treated precisely using the zonotopic abstract domain with our new join operator. Indeed, the implicit affine relations that may not be visible in the assignment of arithmetic expressions, are explicited - and preserved - by the join operation. We implemented this operation in the APRON [17] domain Taylor1+ [7]: some benchmarks show that while the join is still efficient in terms of time and memory (the abstract domain is unchanged, the sole join operation is modified), it is of similar precision as the polyhedric join.

2 Previous results on zonotopic abstract domains

2.1 Affine arithmetic, zonotopes and affine sets

Affine arithmetic is an extension of interval arithmetic on affine forms, first introduced in [2], that takes into account affine correlations between variables. An *affine form* is a formal sum over a set of *noise symbols* ε_i

$$\hat{x} \stackrel{\text{def}}{=} \alpha_0^x + \sum_{i=1}^{n} \alpha_i^x \varepsilon_i,$$

with $\alpha_i^x \in \mathbb{R}$ for all i. Each noise symbol ε_i stands for an independent component of the total uncertainty on the quantity $\hat{x}$, its value is unknown but bounded in [-1,1]; the corresponding coefficient α_i^x is a known real value, which gives the magnitude of that component. The same noise symbol can be shared by several quantities, indicating correlations among them.

The semantics of affine operations is straightforward, they are exact in affine arithmetic. Non affine operations are linearized, and new noise symbols are introduced to handle the approximation term. These new noise symbols are indicated as η_j noise symbols: the ε_i noise symbols model uncertainty in data or parameters, while the η_j noise symbols model uncertainty coming from the analysis. For

instance the multiplication of two affine forms defined on ε_i only (for simplicity of presentation) introduces a new noise symbol, say η_1:

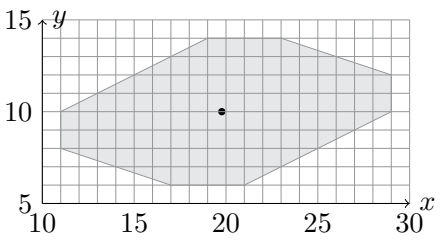
$$\hat{x}\hat{y} = \alpha_0^x\alpha_0^y + \sum_{i=1}^{n}(\alpha_i^x\alpha_0^y + \alpha_i^y\alpha_0^x)\,\varepsilon_i + \left(\sum_{i=1}^{n}|\alpha_i^x\alpha_i^y| + \sum_{i<j}^{n}|\alpha_i^x\alpha_j^y + \alpha_j^x\alpha_i^y|\right)\eta_1.$$

The coefficient of the new perturbation noise symbol η_1 is an over-approximation of the non-linear part of the multiplication.

In what follows, we introduce matrix notations to handle tuples of affine forms. We note $\mathcal{M}(n,p)$ the space of matrices with n lines and p columns of real coefficients. A tuple of affine forms expressing the set of values taken by p variables over n noise symbols ε_i, $1 \leq i \leq n$, can be represented by a matrix $A \in \mathcal{M}(n+1,p)$. We note ${}^tA$ is the transpose of A, and for any vector $e = (e_1,\ldots,e_n) \in \mathbb{R}^n$, $||e||_\infty = \max_{1\leq i\leq n}|e_i|$. We formally define the zonotopic concretization of such tuples by :

Definition 2.1 Let a tuple of affine forms with p variables over n noise symbols be defined by a matrix $A \in \mathcal{M}(n+1,p)$. Its concretization is the zonotope

$$\gamma(A) = \left\{ {}^tA \begin{pmatrix} 1 \\ e \end{pmatrix} \mid e \in \mathbb{R}^n, ||e||_\infty \leq 1 \right\} \subseteq \mathbb{R}^p \ .$$



For instance, for $n = 4$ and $p = 2$, the gray zonotope is the concretization of the affine set $(\hat{x},\hat{y})$, with $\hat{x} = 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4$, $\hat{y} = 10 - 2\varepsilon_1 + \varepsilon_2 - \varepsilon_4$, and ${}^tA = \begin{pmatrix} 20 & -4 & 0 & 2 & 3 \\ 10 & -2 & 1 & 0 & -1 \end{pmatrix}$.

Following the repartition of noise symbols in two sets, we define affine sets as Minkowski sums of a *central* zonotope, $\gamma(C^X)$ and of a *perturbation* zonotope centered on 0, $\gamma(P^X)$. Central zonotopes depend on central noise symbols ε_i, which represent the uncertainty on input values to the program. Perturbation zonotopes depend on perturbation symbols η_j which are created along the interpretation of the program and represent the uncertainty due to operations that are not interpreted exactly.

Definition 2.2 We define an affine set X by the pair of matrices
$X = (C^X, P^X) \in \mathcal{M}(n+1,p) \times \mathcal{M}(m,p)$.
The affine form $X_k = c_{0k}^X + \sum_{i=1}^{n} c_{ik}^X\varepsilon_i + \sum_{j=1}^{m} p_{jk}^X\eta_j$ is the symbolic representation in X of the value of x_k (the kth program variable).

2.2 Geometric and functional orders

Definition 2.3 Let X be an affine set. Its concretization $\gamma(X) \subseteq \mathbb{R}^p$ is the zonotope

$$\gamma(X) = \left\{ {}^tC^X \begin{pmatrix} 1 \\ \varepsilon \end{pmatrix} + {}^tP^X\eta \mid \varepsilon \times \eta \in [-1,1]^{n+m} \right\} .$$

If we were only interested in abstractions of current values of variables, the partial order to consider for proving the correctness of the abstract semantic would be subset inclusion of the concretization, as formalized in Definition 2.3.

However, we consider functional abstractions. Classically, input/output functional abstractions are handled by adding slack variables corresponding to the initial values of the uncertain inputs. Here, we can see the central (noise) symbol ε_i of the affine forms as being these slack variables: an affine set for p variables over n input noise symbols defines a function from $\mathbb{R}^n$ to $\mathbb{R}^p$.

We thus define a pre-order on affine sets [7,11] which formalizes the fact that the central symbols have a specific interpretation as parameterizing the initial values of input arguments to the analyzed program:

Definition 2.4 Let $X = (C^X, P^X)$, $Y = (C^Y, P^Y)$ be two affine sets in $\mathcal{M}(n+1,p) \times \mathcal{M}(m,p)$. We say that $X \leq Y$ iff

$$\forall u \in \mathbb{R}^p,\ ||(C^Y - C^X)u||_1 \leq ||P^Y u||_1 - ||P^X u||_1 \ .$$

where $||X||_1 = \sum_{i=1}^n |X_i|$ is the l_1 norm of vector X.

2.3 Join operator: preliminaries and motivation for the present work

The least upper bound of two affine sets does not exist, there may be incomparable minimal upper bounds. In previous work [10,11], we defined an upper bound for affine sets based on an algorithm that, for each variable taken independently, gives a minimal upper bound in some (most, actually) cases. However, considered globally, this upper bound generally does not give a minimal upper bound: indeed, performing the join operation independently on each variable ignores the relations between variables, and is thus not satisfying.

Let us first recall this join operator over affine sets. For two real numbers α and β, let $\alpha \wedge \beta$ denote their minimum and $\alpha \vee \beta$ their maximum. We define

$$\mathrm{argmin}_{|.|}(\alpha,\beta) = \{\gamma \in [\alpha \wedge \beta, \alpha \vee \beta] \mid |\gamma| \text{ minimal}\}$$

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two intervals. We say that $\boldsymbol{x}$ and $\boldsymbol{y}$ are in generic positions if, whenever $\boldsymbol{x} \subseteq \boldsymbol{y}$, $\inf \boldsymbol{x} = \inf \boldsymbol{y}$ or $\sup \boldsymbol{x} = \sup \boldsymbol{y}$.

Lemma 2.5 *One-dimensional join. [10] Let two affine sets X and Y in $\mathcal{M}(n+1,p) \times \mathcal{M}(m,p)$ such that at least $p-1$ components are equal: for all $l \in [1,p]$, $l \neq k$, $X_l = Y_l$. We define $Z = X \sqcup Y$ by for all $l \in [1,p]$, $l \neq k$, $Z_l = X_l = Y_l$ and $Z_k = X_k \sqcup Y_k$, such that for all $l \in [1,p]$, $l \neq k$, $i = 1 \in [1,n]$, $j \in [1,m]$:*

- $c^Z_{0,k} = mid\,(\gamma(X_k) \cup \gamma(Y_k))$
- $c^Z_{i,k} = argmin_{|.|}(c^X_{i,k}, c^Y_{i,k})$
- $p^Z_{j,k} = argmin_{|.|}(p^X_{j,k}, p^Y_{j,k})$
- $p^Z_{m+k,k} = \sup \gamma(X_k) \cup \gamma(Y_k) - c^Z_{0,k} - \sum_{i=1}^n |c^Z_{i,k}| - \sum_{j=1}^m |p^Z_{j,k}|$
- $p^Z_{m+k,l} = 0$

Then Z is an upper bound of X and Y such that $\gamma(Z_k) = \gamma(X_k) \cup \gamma(Y_k)$. And it is a minimal upper bound whenever $\gamma(X_k)$ and $\gamma(Y_k)$ are in generic positions.

In words, the center of the affine form Z_k is taken as the center of the concretizations of the two affine forms X_k and Y_k. Then the coefficients of Z_k over the noise symbols present in X_k and Y_k are taken as the smallest common dependencies. And finally we add a new noise symbol η_{m+k} to take into account the uncertainty due to the join operation, its coefficient is $p^Z_{m+k,k}$.

This one-dimensional operator can be extended to obtain a p-dimensional join operator, when all dimensions of the affine sets are joined:

Lemma 2.6 *Componentwise join. [11] Let two affine sets X and Y in $\mathcal{M}(n+1, p) \times \mathcal{M}(m, p)$. We define $Z = X \sqcup_C Y$ by: for all $k \in [1, p]$, $Z_k = X_k \sqcup Y_k$. Then Z is an upper bound of X and Y.*

The perturbation added to take into account the uncertainty due to the join is a diagonal block $p \times p$: a new noise symbol η_{m+k} is added for each variable x_k, and these new noise symbols are not shared. This join operator thus loses some relation that may exist between variables.

Let us now consider an example, that motivates a global join operator :

Example 2.7

```
1 float x1 := [1,3];
2 float x2 := [1,3];
3 float x3;
4 if (random()) {
5   x1 = x1 + 2;
6   x2 = x2 + 2; }
7 x3 = x2 − x1;
```

Joining the two branches supposes to join the two affine sets X and Y defined by $(X_1 = 2 + \varepsilon_1, X_2 = 2 + \varepsilon_2)$ and $(Y_1 = 4 + \varepsilon_1, Y_2 = 4 + \varepsilon_2)$. If we apply the component-wise join defined in Lemma 2.5, we obtain Z such that $Z_1 = 3 + \varepsilon_1 + \eta_1$ and $Z_2 = 3 + \varepsilon_2 + \eta_2$, where η_1 and η_2 are two independent new noise symbols. In this case, we do not capture the somehow disjunctive information, that either 0 or 2 is added to both program variables x_1 and x_2, but that it can not be that 0 is added to x_1 and 2 is added to x_2. And we obtain $Z_3 = Z_2 - Z_1 = \varepsilon_2 + \eta_2 - \varepsilon_1 - \eta_1 \in [-4, 4]$.

Now, observe that there is a relation between variables which is true for both branches joined. Since we do not preserve it, the final result is inaccurate. Indeed, it is more than simply a relation between variables, it is a relation between variables and inputs of the program, relations that are captured by our functional abstract domain: we have $x_2 - x_1 = \varepsilon_2 - \varepsilon_1$ in both branches. In order to get a global join operation on X and Y, we can thus use the one-dimensional join operator on

one variable, and naturally deduce the affine form for the second variable by this relation: this gives W such that $W_1 = 3 + \varepsilon_1 + \eta_1$ as previously, and we deduce $W_2 = 3 + \varepsilon_2 + \eta_1$. We thus obtain here a minimal upper bound of X and Y, and can deduce the expected result $W_3 = W_2 - W_1 = \varepsilon_2 - \varepsilon_1 \in [-2, 2]$.

The projection on (x_1, x_2) of the concretization of X, Y, Z and W are represented Figure 1. The component-wise join gives the box $\gamma(Z)$ in which no relation is preserved between Z_1 and Z_2. Whereas the global join gives the zonotope $\gamma(W)$, which is here a minimal upper bound of $\gamma(X)$ and $\gamma(Y)$.

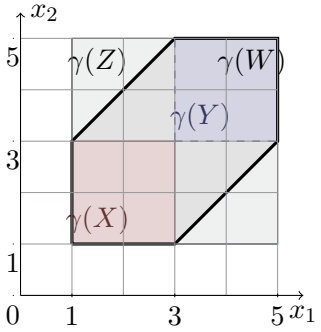


Fig. 1. X, Y and results Z and W of the component-wise and global join operators on Example 2.7

It now remains to formally define these common relations, and to characterize the join operator preserving these relations.

3 Preserving affine relations in join operations

3.1 Affine relations over affine sets

Let us consider an affine set $X = (C^X, P^X)$ over p program variables $x_1, \ldots, x_p$, defined over n central noise symbols $\varepsilon_1, \ldots, \varepsilon_n$ and m perturbation noise symbols $\eta_1, \ldots, \eta_m$. An *affine relation* in X is an affine equation over the p program variables and the n central noise symbols, that holds for any values of the noise symbols, i.e. is given by $\alpha_1, \ldots, \alpha_p, \beta_0, \ldots, \beta_n \in \mathbb{R}$ such that:

$$\sum_{r=1}^{p} \alpha_r x_r = \beta_0 + \sum_{i=1}^{n} \beta_i \varepsilon_i \tag{1}$$

holds for any $\varepsilon = {}^t(\varepsilon_1, \ldots, \varepsilon_n) \in [-1, 1]^n$ and $\eta = {}^t(\eta_1, \ldots, \eta_m) \in [-1, 1]^m$ and $(x_1, \ldots, x_p) = {}^tC^X\varepsilon + {}^tP^X\eta$.

Note that only central noise symbols appear in the equation, and not the perturbation noise symbols. Moreover, we cannot have all the $\alpha_r = 0$ without having all the $\beta_i = 0$ as well. Hence affine relations in an affine set always link the value of at least one x_r with other variables or central noise symbols. It is also well known that the set of affine relations defined in Equation 1, identified with the vector $(\alpha_1, \ldots, \alpha_p, \beta_0, \ldots, \beta_n)$, is a vector space over the reals: sums of any two such vectors, and multiplication of such vectors by any $\lambda \in \mathbb{R}$ still define a valid affine

relation. As a sub-vector space of $\mathbb{R}^{p+n+1}$, it is necessarily of finite dimension, hence generated by a finite basis, i.e. a finite number k of *independent* affine relations. This k is thus necessarily less or equal to p, since the n noise symbols are linearly independent, and there is a constant term.

3.2 Common affine relations

An affine relation common to X and Y is an affine relation in X which is also an affine relation in Y. Consider we have k such affine relations, k being necessarily less or equal to p as previously stated:

$$\sum_{r=1}^{p} \alpha_{l,r} x_r = \beta_{l,0} + \sum_{i=1}^{n} \beta_{l,i} \varepsilon_i, \; \forall l \in \{1 \dots k\}. \tag{2}$$

When the $(x_r)_{1 \leq r \leq p}$ are defined by an affine set $X = (C^X, P^X)$, we rewrite:

$$\textstyle\sum_{r=1}^{p} \alpha_{l,r} x_r = \sum_{i=0}^{n} \left(\sum_{r=1}^{p} \alpha_{l,r} c_{i,r}\right) \varepsilon_i \; + \; \sum_{j=0}^{m} \left(\sum_{r=1}^{p} \alpha_{l,r} p_{j,r}\right) \eta_j, \; \forall l \in \{1, \dots, k\}$$

These relations, being true for every value of the noise symbols ε_i and η_j, imply that for all $1 \leq l \leq k$, $0 \leq i \leq n$ and $1 \leq j \leq m$, we have:

$$\sum_{r=1}^{p} \alpha_{l,r} c_{i,r} = \beta_{l,i} \tag{3}$$

and

$$\sum_{r=1}^{p} \alpha_{l,r} p_{j,r} = 0 \tag{4}$$

Example 3.1 Consider again the two affine sets joined in Example 2.7:

$$\begin{aligned} X_1 &= 2 + \varepsilon_1 \\ X_2 &= 2 + \varepsilon_2 \end{aligned} \quad \text{and} \quad \begin{aligned} Y_1 &= 4 + \varepsilon_1 \\ Y_2 &= 4 + \varepsilon_2 \end{aligned}$$

We want the affine relations common to X and Y, of the form: $\alpha_1 x_1 + \alpha_2 x_2 = \beta_0 + \beta_1 \varepsilon_1 + \beta_2 \varepsilon_2$ that hold for all $(\varepsilon_1, \varepsilon_2) \in [-1, 1]$. Substituting X and Y in this relation yields $\beta_0 = 2\alpha_1 + 2\alpha_2 = 4\alpha_1 + 4\alpha_2$, $\beta_1 = \alpha_1$ and $\beta_2 = \alpha_2$. The solutions can be parameterized by a $\lambda \in \mathbb{R}$, they are of the form $\beta_0 = 0$, $\alpha_1 = \beta_1 = \lambda$, $\alpha_2 = \beta_2 = -\lambda$. For instance we can choose $x_2 - x_1 = \varepsilon_2 - \varepsilon_1$.

3.3 Reduction to row-echelon form

Up to a renumbering of the variables x_1,..., x_p, we can always suppose the k independent affine relations common to X and Y are of the form:

$$x_i = R_i(x_{i+1}, \dots, x_p, \varepsilon_1, \dots, \varepsilon_n), \tag{5}$$

where R_i is an affine function, for all $i \in [1..k]$.

This can be shown as follows. We first consider the matrix $M = {}^t(D_1, \dots, D_k)$ whose ith row is the vector $D_i \in \mathbb{R}^{p+n+1}$ representing the ith of the k independent affine relations common to X and Y. This matrix can be written $M = (M'|M'')$ with M' a $k \times k$ matrix (only the coefficient of the x_i appear) and M'' is the remaining block of size $k \times (p + n + 1 - k)$.

Now, M' has an LU decomposition (see e.g. [1]) $M' = LU$, where L is lower triangular and ones on the diagonal, U is upper triangular. Then $(U|L^{-1}M'')$ is a matrix of affine relations common to X and Y as well: indeed, $(U|L^{-1}M'') = L^{-1}M$ and thus $(U|L^{-1}M'')u = 0$ is equivalent to $Mu = 0$. Now, as the rank of M is k, as it is made of k independent affine relations, U has exactly k non-null elements on the diagonal, giving k independent relations of the form defined in Equation 5.

3.4 Definition of a join operator

Suppose we have two affine sets X and Y defining p variables, our new join operator is defined by Algorithm 1, that we prove correct, and even optimal in some cases, in this section (cf. Proposition 2.6). In what follows, we note $X_{>k}$ the affine set obtained by the projection of $X \in \mathcal{M}(n+1,p)\times\mathcal{M}(m,p)$ on its $p-k$ last components $x_{k+1},\ldots,x_p$.

Algorithm 1 Algorithm for the global join operator

Inputs are $X = (C^X, P^X)$, $Y = (C^Y, P^Y)$, output is Z
Form the system of equations in $\alpha_{l,r}$, $\beta_{l,i}$, for $1 \leq l \leq p$, $0 \leq i \leq n$, $1 \leq j \leq m$:

$$\begin{cases} \sum_{r=1}^{p} \alpha_{l,r} c^X_{i,r} = \sum_{r=1}^{p} \alpha_{l,r} c^Y_{i,r} = \beta_{l,i} \\ \sum_{r=1}^{p} \alpha_{l,r} p^X_{j,r} = \sum_{r=1}^{p} \alpha_{l,r} p^Y_{j,r} = 0 \end{cases}$$

Solve this system to get k independent affine relations, $i = 1,\ldots,k$, using LU decomposition: $x_i = R_i(x_{i+1},\ldots,x_p,\varepsilon_1,\ldots,\varepsilon_n)$
Define $Z = X \sqcup Y$ as:
For rows strictly greater than k: $Z_{>k} = X_{>k} \sqcup_C Y_{>k}$ (cf. Lemma 2.6)
For rows less than k:
for $i = k$ downto 1 **do**
  $Z_i = R_i(Z_{i+1},\ldots,Z_p,\varepsilon_1,\ldots,\varepsilon_n)$
end for

Basically, the algorithm works as follows: we determine the k independent affine relations between the variables and the input noise symbols ε_i, that they have in common. We express them as in Equation 2. We thus can choose among the p variables, $p-k$ variables, on which we use the component-wise join of Lemma 2.6, and from there reconstruct an upper bound for the p variables using Equation 5.

The global join of Algorithm 1 is therefore mathematically defined as :

Definition 3.2 Global join. Let X and Y be two affine sets in $\mathcal{M}(n+1,p) \times \mathcal{M}(m,p)$, which have in common k independent affine relations:
for all $i \in [1,k]$, $x_i = R_i(x_{i+1},\ldots,x_p,\varepsilon_1,\ldots,\varepsilon_n)$. We define $Z = X \sqcup_G Y$ by $Z_{>k} = X_{>k} \sqcup_C Y_{>k}$ and for all $i \in k,\ldots,1$, $Z_i = R_i(Z_{i+1},\ldots,Z_p,\varepsilon_1,\ldots,\varepsilon_n)$.

Note that in Definition 3.2, the operations are ordered: first computation of $Z_{>k}$, then reconstruction of Z_k to Z_1. Due to the row-echelon reduction and the reconstruction, the worst-case time complexity of this global join is $O(n^3 + n^2 p)$,

whereas the component-wise join's complexity is only $O(np)$. However, we introduce less new noise symbols and we are more precise, so this new join operation may speed up fixpoint computations as shown in Section 4.

Proposition 3.3 now proves the correctness of Algorithm 1, as well as optimality in some cases (in particular, when $k = p - 1$, because of the optimality of the uni-dimensional join in the generic case) :

Proposition 3.3 *$Z = X \sqcup_G Y$ is an upper bound of X and Y, and if $Z_{>k}$ is a minimal upper bound of $X_{>k}$ and $Y_{>k}$, then Z is a minimal upper bound of X and Y.*

A lemma (needed to prove Proposition 3.3) states that relations of Equation 2 are compatible with the functional order. This property shows the necessity to preserve affine input/output relations.

Lemma 3.4 *Let $X = (C^X, P^X)$ and $Y = (C^Y, P^Y)$ be two affine sets such that:*

(i) *$X \leq Y$,*

(ii) *Y satisfies the k relations of Equation 2*

Then X satisfies these k relations.

This join operator is indirectly linked to Karr's algorithm [14,16], as it computes and preserves affine equalities that Karr's algorithm would infer if applied functionally, that is equalities between variables and inputs (our noise symbols).

4 Experiments

The APRON [17] library implements different abstract domains, including the affine sets (Taylor1+ domain). We implemented this new join operator in the Taylor1+ abstract domain. We present some results on programs analyzed with Interproc, a static analyzer which allows us to select any APRON abstract domain. First, we focus on comparisons of results of join operators with Taylor1+ using the standard (component-wise) and the global joins. We then show the relative performance of all classical domains and our domains on an example with different discretization steps. We refer to [7,8] for examples which really aim at demonstrating the performance of zonotopic abstract domains in general.

We consider three examples (Fig 5) that illustrate some constructions commonly found in classical programs. *Loop counter* (Fig 2) is a program in which the value of a variable x depends indirectly on the value of the loop counter. With the global join operator, we infer that $x - i = 2 + 2\varepsilon_1$ and the analysis reaches a fix-point without the help of a widening operator. With the classical join operator, the analysis cannot terminate without widening, and terminates with the disappointing $x \in [0, \infty]$. *If branches* (Fig 3) show what may happen when we join the two branches of an conditional structure that encodes a computation on variables x and y, depending on the current mode (m, considered as a value 0 or 1) of the program. In this example, the difference is more subtle since the results of both join operations

have the same interval concretization. However, the global join is better since it introduces only one new noise symbol instead of three. *Linear recurrence* (Fig 4) computes a sequence of couples that converges to $(14, 14)$. Without the global join, the static analysis with zonotopic abstract domain cannot find out this limit.

```
1  float x=[0,4];
2  int i=0;
3  while (i ≤ 5) {
4      i++;
5      x++;}
```

Fig. 2. Loop counter

```
1  float x=[0,4];
2  float y=[0,4];
3  bool m = brandom();
4  if (m) {
5      x++; y++;
6      m= not(m);}
7    else {
8      x−−; y−−;
9      m= not(m);}
```

Fig. 3. If branches

```
1  float x=12;
2  float x1=12;
3  float y=16;
4  float y1=16;
5  while (true) {
6    x=x1;
7    y=y1;
8    x1=3∗x/4 + y/4;
9    y1=x/4 + 3∗ y/4;}
```

Fig. 4. Linear recurrence

example	standard join	global join	relation
loop counter	$i \in [0, 6]$ $x \in [0, \infty]$	$i = 3 + 3\eta_2 \in [0, 6]$ $x = 5 + 2\varepsilon_1 + 3\eta_2 \in [0, 10]$	$x - i = 2 + 2\varepsilon_1$
if branches	$x = 2 + 2\varepsilon_1 + \eta_1 \in [-1, 5]$ $y = 2 + 2\varepsilon_2 + \eta_2 \in [-1, 5]$ $m = 0.5 + 0.5\eta_3 \in [0, 1]$	$x = 2 + 2\varepsilon_1 + \eta_1 \in [-1, 5]$ $y = 2 + 2\varepsilon_2 + \eta_1 \in [-1, 5]$ $m = 0.5 - 0.5\eta_1 \in [0, 1]$	$x + 2m = 3 + 2\varepsilon_1$ $y + 2m = 3 + 2\varepsilon_2$
linear recurrence	$x, x_1 \in [12, 16]$ $y, y_1 \in [12, 16]$	$x, x_1 \in [12, 14]$ $y, y_1 \in [14, 16]$	$x_1 + y_1 = 28$ $x + y = 28$

Fig. 5. Comparison between standard join and global join

```
1  float f(float x) {
2    return 2∗x−3; }

4  float g(float x) {
5    return −x+5; }

7  int main() {
8    int i;
9    float x,y,z,t,u,v;

10   y = f(0); z = g(0);
11   u = f(.75); v = g(.25);
12   for (i=1; i<=N; i++) {
13       x=[0,((float)i)/N];
14       y=f(x); z=g(x);
15       u=f(v); v=g(u)/2; }
16   t=y+2∗z;
17   return 0; }
```

Fig. 6. Program to compare abstract domains of APRON

Finally, we consider a function to challenge the performance and the precision of the APRON abstract domains (Fig. 6). We iterate this function by a Kleene iteration, without widening. The final value does not depend on the parameter N; increasing N only increases the number of join operations performed. We compared four abstract domains of APRON (boxes, octagons, convex polyhedra, Taylor1+)

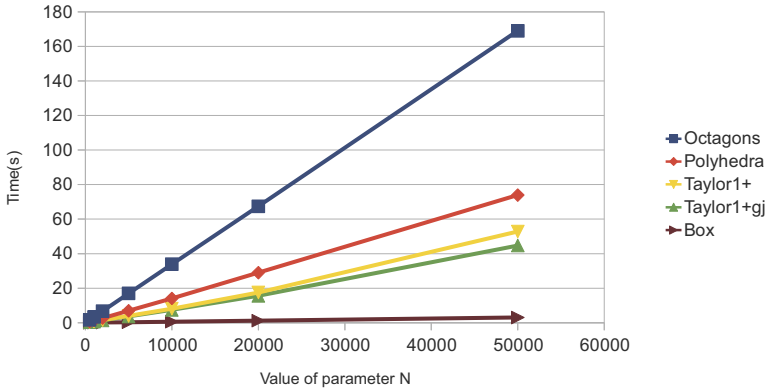


Fig. 7. Comparison of different abstract domains

to the domain Taylor1+ with global join (Taylor1+gj). Taylor1+ with global join provides an analysis slightly faster than classical Taylor1+, and up to 40 percent faster than polyhedra, and even 74 percent faster than octagons (Figure 7). The global join operator on Taylor1+ allows to prove the exact invariant $t = 7$, that only polyhedra find as well. Boxes find $t \in [5, 9]$, octagons $t \in [5.5, 8.5]$, Taylor1+ with standard join $t \in [5, 9]$.

5 Conclusion

In this paper, we have proposed a new join operator of zonotopic abstract domains. This addresses the main drawback of the domain, which apart from the join operator, was known for providing precise and fast analyses. This join operator stays in the line of a functional (input/output) analysis, by discovering and preserving some common affine relations holding between program variables and inputs of the program. It also allows to discover some useful properties, that for instance link the current program variables values to loops counters, or even make apparent some disjunctive information. We showed that while improving the accuracy compared to the previous join operation of [7], we still keep a very fast operation. We believe that this approach generalizes easily to the case of constrained affine sets, as introduced in [8]. Future work includes the case of inexact common affine relations, for instance to handle the case when they approximate non-linear relations.

References

[1] Rob Beezer. *A First Course in Linear Algebra.* available at http://linear.ups.edu/online.html, 2006.

[2] J. L. D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. *Proceedings of SIBGRAPI*, 1993.

[3] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE static analyzer. In *ESOP'05*, pages 21–30, 2005.

[4] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL'78*, pages 84–96. ACM Press, 1978.

[5] D. Delmas, E. Goubault, S. Putot, J. Souyris, K. Tekkal, and F. Védrine. Towards an industrial use of FLUCTUAT on safety-critical avionics software. In *FMICS'09, LNCS 5825*, pages 53–69. Springer-Verlag, 2009.

[6] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *CAV'11*, LNCS. Springer, 2011.

[7] K. Ghorbal, E. Goubault, and S. Putot. The zonotope abstract domain Taylor1+. In *CAV'09, LNCS 5643*, pages 627–633. Springer-Verlag, 2009.

[8] K. Ghorbal, E. Goubault, and S. Putot. A logical product approach to zonotope intersection. In *CAV'10*, LNCS 6174, pages 212–226, 2010.

[9] A. Girard. Reachability of uncertain linear systems using zonotopes. In *HSCC'05, LNCS 3414*, pages 291–305. Springer-Verlag, 2005.

[10] E. Goubault and S. Putot. Perturbed affine arithmetic for invariant computation in numerical program analysis. *CoRR*, abs/0807.2961, available at http://arxiv.org/abs/0807.2961, 2008.

[11] E. Goubault and S. Putot. A zonotopic framework for functional abstractions. *CoRR*, abs/0910.1763, available at http://arxiv.org/abs/0910.1763, 2009.

[12] Eric Goubault and Sylvie Putot. Static analysis of finite precision computations. In *VMCAI'11, LNCS 6530*, pages 232–247, 2011.

[13] Leonidas J. Guibas, An Nguyen, and Li Zhang. Zonotopes as bounding volumes. In *Proceedings of 14 t h Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[14] Michael Karr. Affine relationships among variables of a program. *Acta Inf.*, 6:133–151, 1976.

[15] A. Miné. A new numerical abstract domain based on difference-bound matrices. In *Proceedings of the Second Symposium on Programs as Data Objects, LNCS 2053*, pages 155–172. Springer-Verlag, 2001.

[16] Markus Müller-Olm and Helmut Seidl. A note on Karr's algorithm. In *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 1016–1028, 2004.

[17] APRON Project. Numerical abstract domain library, 2007. http://apron.cri.ensmp.fr.

[18] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *VMCAI'05, LNCS 3385*, pages 25–41, 2005.