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

Eric Goubault, Tristan Le Gall and
Sylvie Putot
CEA, LIST - LMeASI
NSAD'12

Static Analysis of Numerical Programs

- Goal : to find numerical invariants, to give an upper bound for numerical errors
- Problems :
  - infinite domains $\Rightarrow$ symbolic representation
  - precision, difference between real numbers arithmetics and floating-point arithmetics
  - infinite loops, numerical drift (e.g. Patriot missile)

Numerical Abstract Domains

- Classical ones : Intervals, convex polyhedra
- Recent ones : Octagons, linear templates
- In Fluctuat : Affine sets (zonotopes)

Static Analysis of Numerical Programs

- Goal : to find numerical invariants, to give an upper bound for numerical errors
- Problems :
  - infinite domains $\Rightarrow$ symbolic representation
  - precision, difference between real numbers arithmetics and floating-point arithmetics
  - infinite loops, numerical drift (e.g. Patriot missile)

Numerical Abstract Domains

- Classical ones : Intervals, convex polyhedra
- Recent ones : Octagons, linear templates
- In Fluctuat : Affine sets (zonotopes)

We present a new, accurate join operator for zonotopes

Outline

Presentation of the abstract domain

A new join operator

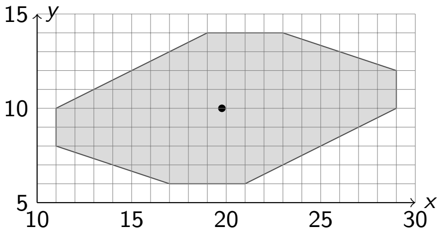
Experiments

Zonotopes (1) : Symbolic Representation and Concretization

Symbolic representation

- Each variable = linear sum of noise symbols : $\hat{x} = 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4$
- Noise symbols are shared variables, whose range is $[-1, 1]$
- Alternative definition : Minkowski sum of vectors defined by the coefficients of the noise symbols

An affine set and its concretization



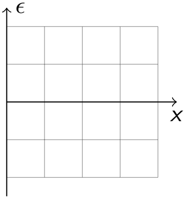
The gray zonotope is the concretization of the affine set $(\hat{x}, \hat{y})$, with $\hat{x} = 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4$, $\hat{y} = 10 - 2\varepsilon_1 + \varepsilon_2 - \varepsilon_4$, and ${}^tA = \begin{pmatrix} 20 & -4 & 0 & 2 & 3 \\ 10 & -2 & 1 & 0 & -1 \end{pmatrix}$

Functional Order, Augmented Space

- Partial order on affine sets is a functional order

Example

$\hat{x} = 2 + \epsilon$ and $\hat{x} = 2 - \epsilon$ (concretization : $[1, 3]$)

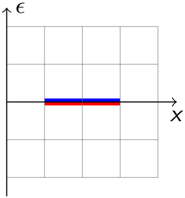


Functional Order, Augmented Space

- Partial order on affine sets is a functional order
- Functional order $\neq$ geometrical order of the concretization in $\mathbb{R}^p$

Example

$\hat{x} = 2 + \epsilon$ and $\hat{x} = 2 - \epsilon$ (concretization : $[1, 3]$)

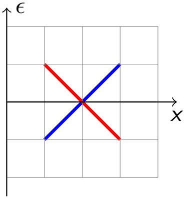


Functional Order, Augmented Space

- Partial order on affine sets is a functional order
- Functional order $\neq$ geometrical order of the concretization in $\mathbb{R}^p$
- Functional order $=$ geometrical order in augmentend space $\mathbb{R}^{p+n}$

Example

$\hat{x} = 2 + \epsilon$ and $\hat{x} = 2 - \epsilon$ (concretization : $[1, 3]$)



Consider two affines sets $\hat{x} = 2 + 3\epsilon_1 - 2\epsilon_2$ and $\hat{y} = 3 + 2\epsilon_2$

Addition $x + y$

- Exact operation

$$\widehat{x + y} = 5 + 3\epsilon_1$$

Multiplication $x \times y$

- Exact operation

$$\widehat{x \times y} = 6 + 9\epsilon_1 + (4 - 6)\epsilon_2 + 6\epsilon_1\epsilon_2 - 4\epsilon_2^2$$

Consider two affines sets $\hat{x} = 2 + 3\epsilon_1 - 2\epsilon_2$ and $\hat{y} = 3 + 2\epsilon_2$

Addition $x + y$

- Exact operation

$$\widehat{x+y} = 5 + 3\epsilon_1$$

Multiplication $x \times y$

- Exact operation

$$\widehat{x \times y} = 6 + 9\epsilon_1 + (4-6)\epsilon_2 + 6\epsilon_1\epsilon_2 - 4\epsilon_2^2$$

- Second-order terms range in $[-10, 2.25] = -3.875 + 6.125\eta_1$

$$\widehat{x \times y} = 2.125 + 9\epsilon_1 - 2\epsilon_2 + 6.125\eta_1$$

Advantages

- Relational lattice, cheap linear assignments
- Non-linear assignments (Taylor, 1st order)

Drawbacks

- Meet
- Join

Advantages

- Relational lattice, cheap linear assignments
- Non-linear assignments (Taylor, 1st order)

Drawbacks / improvements

- Meet : constrained affine sets
- Join

Advantages

- Relational lattice, cheap linear assignments
- Non-linear assignments (Taylor, 1st order)

Drawbacks / improvements

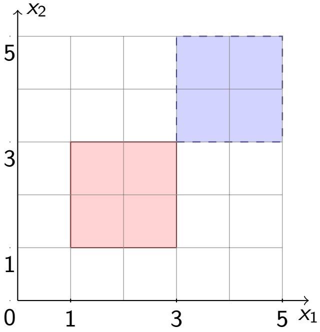
- Meet : constrained affine sets
- Join : global join

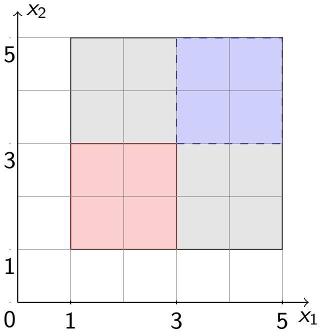
Example of join

```
double x1 := [1,3] ;
double x2 := [1,3] ;
double x3 ;
if (random()) {
    x1 = x1 + 2 ;
    x2 = x2 + 2 ; }
x3 = x2 - x1 ;
```

Affine sets :

$$\begin{array}{rcl} \hat{x_1} & = & 2+\epsilon_1 \\ \hat{x_2} & = & 2+\epsilon_2 \\ \hat{x_3} & = & \top \end{array} \quad \text{and} \quad \begin{array}{rcl} \hat{x_1} & = & 4+\epsilon_1 \\ \hat{x_2} & = & 4+\epsilon_2 \\ \hat{x_3} & = & \top \end{array}$$

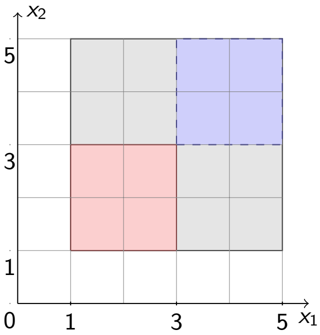

x_2
5
3
1
0
1
3
5
x_1



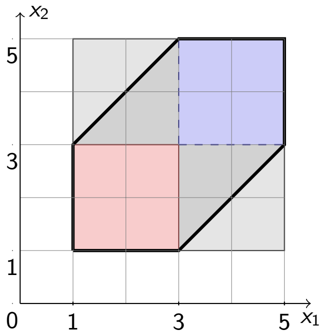
- Componentwise join (one dimension at a time)

$$\begin{array}{rcl} \hat{x_1} & = & 3 + \epsilon_1 + \eta_1 \\ \hat{x_2} & = & 3 + \epsilon_2 + \eta_2 \\ \hat{x_3} & = & \top \end{array}$$

Example (2)



- Componentwise join (one dimension at a time)
- Common affine relation : $x_1 - x_2 = \epsilon_1 - \epsilon_2$



- Componentwise join (one dimension at a time)
- Common affine relation : $x_1 - x_2 = \epsilon_1 - \epsilon_2$
- Global join

Goal : to preserve affine relations

- Two affine sets X and Y, p variables $x_1 \dots x_p$, $n+1$ noise symbols $\varepsilon_0, \dots, \varepsilon_n$
- An affine relation : $\alpha_1 x_1 + \cdots + \alpha_p x_p = \beta_0 \varepsilon_0 + \beta_1 \varepsilon_1 + \cdots + \beta_n \varepsilon_n$
- Our goal : to find an upper bound Z that preserves common affine relations

Goal : to preserve affine relations

- Two affine sets X and Y, p variables $x_1 \dots x_p$, $n+1$ noise symbols $\varepsilon_0, \dots, \varepsilon_n$
- An affine relation : $\alpha_1 x_1 + \dots + \alpha_p x_p = \beta_0 \varepsilon_0 + \beta_1 \varepsilon_1 + \dots + \beta_n \varepsilon_n$
- Our goal : to find an upper bound Z that preserves common affine relations

Issues

1. How to discover common affine relations ?
2. How to reduce the size of the problem ?
3. How to rebuild the affine sets with the help of the affine relations ?

Augmented space

- Program variables + noise symbols : vector space, dimension $p + n + 1$
- Functional order = geometrical order
- A relation defines an hyperplane containing the zonotope.

Augmented space

- Program variables + noise symbols : vector space, dimension $p + n + 1$
- Functional order = geometrical order
- A relation defines an hyperplane containing the zonotope.

General algorithm

Assume we have k relations, defining the variables $x_1, \ldots, x_k$, we compute $X \sqcup_G Y$:

1. Existential quantification : $X_{>k}$ and $Y_{>k}$ (elimination of $x_1, \ldots, x_k$)
2. Componentwise join $Z_{>k} = X_{>k} \sqcup Y_{>k}$
3. Reconstruction (intersection with hyperplanes)

Any relation true for both X and Y is also true for Z.

Algorithm to find affine relations

1. The value of each variable is replaced by its expression (linear sum of noise symbol)
2. The coefficients of noise symbols must be equal in both affine sets X and Y
3. One equation per noise symbol, then we solve them by a Gauss reduction to obtain the coefficients α_i , then the coefficients β_i
4. Solutions belong to a vector space (finite dimension)

Example

Affine sets X and Y :

$$\begin{array}{lcl} x_1 & = & 2+\epsilon_1 \\ x_2 & = & 2+\epsilon_2 \\ x_3 & = & \top \end{array} \quad \text{and} \quad \begin{array}{lcl} x_1 & = & 4+\epsilon_1 \\ x_2 & = & 4+\epsilon_2 \\ x_3 & = & \top \end{array}$$

We are looking for a relation :

$$\alpha_1 x_1 + \alpha_2 x_2 \;=\; \beta_0 + \beta_1 \epsilon_1 + \beta_2 \epsilon_2$$

Affine sets X and Y :

$$\begin{array}{lll} x_1 & = & 2+\epsilon_1 \\ x_2 & = & 2+\epsilon_2 \\ x_3 & = & \top \end{array} \quad \text{and} \quad \begin{array}{lll} x_1 & = & 4+\epsilon_1 \\ x_2 & = & 4+\epsilon_2 \\ x_3 & = & \top \end{array}$$

Example (cont.)

1. We replace x_1 and x_2 by their expressions :

$$\alpha_1(2+\epsilon_1)+\alpha_2(2+\epsilon_2) \;=\; \beta_0+\beta_1\epsilon_1+\beta_2\epsilon_2$$

and :

$$\alpha_1(4+\epsilon_1)+\alpha_2(4+\epsilon_2) \;=\; \beta_0+\beta_1\epsilon_1+\beta_2\epsilon_2$$

Affine sets X and Y :

$$\begin{array}{rcl} x_1 & = & 2+\epsilon_1 \\ x_2 & = & 2+\epsilon_2 \\ x_3 & = & \top \end{array} \quad \text{and} \quad \begin{array}{rcl} x_1 & = & 4+\epsilon_1 \\ x_2 & = & 4+\epsilon_2 \\ x_3 & = & \top \end{array}$$

Example (cont.)

1. We replace x_1 and x_2 by their expressions :

   $$\alpha_1(2+\epsilon_1)+\alpha_2(2+\epsilon_2) \;=\; \beta_0+\beta_1\epsilon_1+\beta_2\epsilon_2$$

   and :

   $$\alpha_1(4+\epsilon_1)+\alpha_2(4+\epsilon_2) \;=\; \beta_0+\beta_1\epsilon_1+\beta_2\epsilon_2$$

2. The coefficients of the noise symbols must be equal ; we get the equations : $2\alpha_1+2\alpha_2=4\alpha_1+4\alpha_2$, and $\beta_0=0$, $\beta_1=\alpha_1$, $\beta_2=\alpha_2$

Affine sets X and Y :

$$\begin{array}{lcl} x_1 & = & 2+\epsilon_1 \\ x_2 & = & 2+\epsilon_2 \\ x_3 & = & \top \end{array} \quad \text{and} \quad \begin{array}{lcl} x_1 & = & 4+\epsilon_1 \\ x_2 & = & 4+\epsilon_2 \\ x_3 & = & \top \end{array}$$

Example (cont.)

1. We replace x_1 and x_2 by their expressions :

$$\alpha_1(2+\epsilon_1)+\alpha_2(2+\epsilon_2) \;=\; \beta_0+\beta_1\epsilon_1+\beta_2\epsilon_2$$

   and :

$$\alpha_1(4+\epsilon_1)+\alpha_2(4+\epsilon_2) \;=\; \beta_0+\beta_1\epsilon_1+\beta_2\epsilon_2$$

2. The coefficients of the noise symbols must be equal ; we get the equations : $2\alpha_1+2\alpha_2=4\alpha_1+4\alpha_2$, and $\beta_0=0$, $\beta_1=\alpha_1$, $\beta_2=\alpha_2$
3. Example of solution : $\alpha_1=1$, $\alpha_2=-1, \beta_0=0, \beta_1=1, \beta_2=-1$

Discovery of common affine relations (2)

Affine sets X and Y :

$$\begin{array}{rcl} x_1 & = & 2+\epsilon_1 \\ x_2 & = & 2+\epsilon_2 \\ x_3 & = & \top \end{array} \quad \text{and} \quad \begin{array}{rcl} x_1 & = & 4+\epsilon_1 \\ x_2 & = & 4+\epsilon_2 \\ x_3 & = & \top \end{array}$$

Example (cont.)

1. We replace x_1 and x_2 by their expressions :

$$\alpha_1(2+\epsilon_1)+\alpha_2(2+\epsilon_2) \;=\; \beta_0+\beta_1\epsilon_1+\beta_2\epsilon_2$$

   and :

$$\alpha_1(4+\epsilon_1)+\alpha_2(4+\epsilon_2) \;=\; \beta_0+\beta_1\epsilon_1+\beta_2\epsilon_2$$

2. The coefficients of the noise symbols must be equal ; we get the equations : $2\alpha_1+2\alpha_2=4\alpha_1+4\alpha_2$, and $\beta_0=0$, $\beta_1=\alpha_1$, $\beta_2=\alpha_2$
3. Example of solution : $\alpha_1=1$, $\alpha_2=-1, \beta_0=0, \beta_1=1, \beta_2=-1$
4. Relation : $x_1=x_2+\varepsilon_1-\varepsilon_2$

Algorithm

Assume we have k relations, defining the variables $x_1, \dots, x_k$. We compute $X \sqcup_G Y$

1. Existential quantification : $X_{>k}$ and $Y_{>k}$ (elimination of $x_1, \dots, x_k$)
2. Componentwise join $Z_{>k} = X_{>k} \sqcup Y_{>k}$
3. Reconstruction

Example

Relation $x_1 = x_2 + \epsilon_1 - \epsilon_2$.

$$X = \begin{cases} \hat{x}_1 = 2 + \epsilon_1 \\ \hat{x}_2 = 2 + \epsilon_2 \\ \hat{x}_3 = \top \end{cases} \quad Y = \begin{cases} \hat{x}_1 = 4 + \epsilon_1 \\ \hat{x}_2 = 4 + \epsilon_2 \\ \hat{x}_3 = \top \end{cases}$$

Algorithm

Assume we have k relations, defining the variables $x_1, \ldots, x_k$. We compute $X \sqcup_G Y$

1. Existential quantification : $X_{>k}$ and $Y_{>k}$ (elimination of $x_1, \ldots, x_k$)
2. Componentwise join $Z_{>k} = X_{>k} \sqcup Y_{>k}$
3. Reconstruction

Example

Relation $x_1 = x_2 + \epsilon_1 - \epsilon_2$.

$$X_{>k} = \begin{cases} \hat{x}_1 = \top \\ \hat{x}_2 = 2 + \epsilon_2 \\ \hat{x}_3 = \top \end{cases} \quad Y_{>k} = \begin{cases} \hat{x}_1 = \top \\ \hat{x}_2 = 4 + \epsilon_2 \\ \hat{x}_3 = \top \end{cases}$$

Algorithm

Assume we have k relations, defining the variables $x_1, \ldots, x_k$. We compute $X \sqcup_G Y$

1. Existential quantification : $X_{>k}$ and $Y_{>k}$ (elimination of $x_1, \ldots, x_k$)
2. Componentwise join $Z_{>k} = X_{>k} \sqcup Y_{>k}$
3. Reconstruction

Example

Relation $x_1 = x_2 + \epsilon_1 - \epsilon_2$.

$$X_{>k} = \begin{cases} \hat{x}_1 = \top \\ \hat{x}_2 = 2 + \epsilon_2 \\ \hat{x}_3 = \top \end{cases} \quad Y_{>k} = \begin{cases} \hat{x}_1 = \top \\ \hat{x}_2 = 4 + \epsilon_2 \\ \hat{x}_3 = \top \end{cases} \quad Z_{>k} = \begin{cases} \hat{x}_1 = \top \\ \hat{x}_2 = 3 + \epsilon_2 + \eta \\ \hat{x}_3 = \top \end{cases}$$

Algorithm

Assume we have k relations, defining the variables $x_1, \dots, x_k$. We compute $X \sqcup_G Y$

1. Existential quantification : $X_{>k}$ and $Y_{>k}$ (elimination of $x_1, \dots, x_k$)
2. Componentwise join $Z_{>k} = X_{>k} \sqcup Y_{>k}$
3. Reconstruction

Example

Relation $x_1 = x_2 + \epsilon_1 - \epsilon_2$.

$$X = \begin{cases} \hat{x}_1 = 2 + \epsilon_1 \\ \hat{x}_2 = 2 + \epsilon_2 \\ \hat{x}_3 = \top \end{cases} \quad Y = \begin{cases} \hat{x}_1 = 4 + \epsilon_1 \\ \hat{x}_2 = 4 + \epsilon_2 \\ \hat{x}_3 = \top \end{cases} \quad Z = \begin{cases} \hat{x}_1 = 3 + \epsilon_1 + \eta \\ \hat{x}_2 = 3 + \epsilon_2 + \eta \\ \hat{x}_3 = \top \end{cases}$$

Theorem

$Z = X \sqcup_G Y$ 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.

Theorem

$Z = X \sqcup_G Y$ 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.

Remarks

- Any relation true for X and Y is also true for Z
- The componentwise join $Z_{>k} = X_{>k} \sqcup Y_{>k}$ is a minimal upper bound if $k = p - 1$
- We can do the same for the widening

Program 1

```
double x=[0,4];
int i=0;
while i ≤ 5 {
    i++;
    x++;}
```

- Issue : (lack of) explicit relation between x and i

Program 1

```
double x=[0,4];
int i=0;
while i ≤ 5 {
    i++;
    x++;}
```

- Issue : (lack of) explicit relation between x and i
- Componentwise join : no convergence (without widening)

Program 1

```
double x=[0,4];
int i=0;
while i ≤ 5 {
    i++;
    x++;}
```

- Issue : (lack of) explicit relation between x and i
- Componentwise join : no convergence (without widening)
- Global join : loop invariant $x - i = 2 + 2\epsilon_1$ (thus $x \in [0, 10]$)

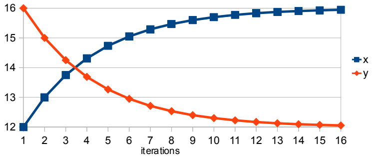
Program 2

```
double x=12;
double x1=12;
double y=16;
double y1=16;
while (true) {
    x=x1;
    y=y1;
    x1=3*x/4 + y/4;
    y1=x/4 + 3* y/4;}
```

Program 2

```
double x=12;
double x1=12;
double y=16;
double y1=16;
while (true) {
    x=x1;
    y=y1;
    x1=3*x/4 + y/4;
    y1=x/4 + 3* y/4;}
```

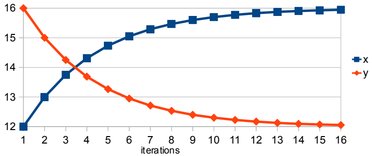
componentwise join



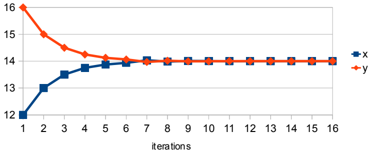
Program 2

```
double x=12;
double x1=12;
double y=16;
double y1=16;
while (true) {
    x=x1;
    y=y1;
    x1=3*x/4 + y/4;
    y1=x/4 + 3* y/4;}
```

componentwise join



global join



Experiments (3) : Benchmarks

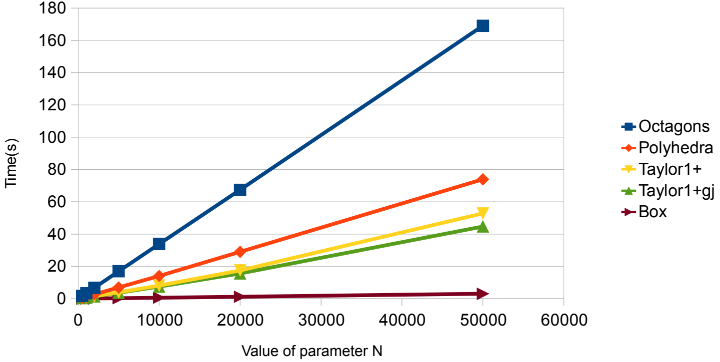
Program 3

```
double f(double x) {
   return 2*x-3 ; }

double g(double x) {
   return -x+5 ; }

int main() {
   y = f(0) ; z = g(0) ;
   u = f(.75) ; v = g(.25) ;
   for (i=1 ; i¡=N ; i++) {
      x=[0,((double)i)/N] ;
      y=f(x) ; z=g(x) ;
      u=f(v) ; v=g(u)/2 ; }
   t=y+2*z ; return 0 ; }
```

Increasing N increases the number of operations, but does not change the result.



Exact result : only polyhedra and zonotopes with global join

Summary

- A nice improvement of the join operator for zonotopes
- Implementation (APRON)

Ongoing work

- Implementation (Fluctuat)
- Imprecise relations
- Policy Iteration