

Relational Numerical Abstract Domains

MPRI 2–6: Abstract Interpretation,
application to verification and static analysis

Antoine Miné

Year 2017–2018

Course 04

4 October 2017

Outline

- The need for relational domains
- Presentation of a few **relational numerical abstract domains**
 - **linear equality** domain
 - **polyhedra** domain
 - weakly relational domains: **zones**, **octagons**
- Bibliography

Shortcomings of non-relational domains

Accumulated loss of precision

Non-relation domains cannot represent variable **relationships**

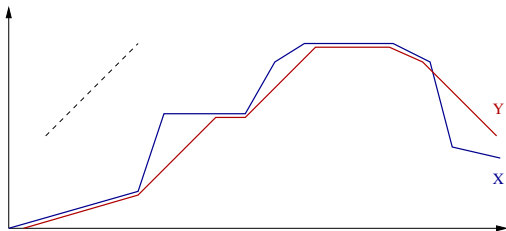
Rate limiter

```

Y ← 0; while 1=1 do
  X ← [-128,128]; D ← [0,16];
  S ← Y; Y ← X; R ← X - S;
  if R ≤ -D then Y ← S - D fi;
  if R ≥ D then Y ← S + D fi
done

```

X: input signal
 Y: output signal
 S: last output
 R: delta $Y - S$
 D: max. allowed for $|R|$



Accumulated loss of precision

Non-relation domains cannot represent variable **relationships**

Rate limiter

```

Y ← 0; while ● 1=1 do
  X ← [-128,128]; D ← [0,16];
  S ← Y; Y ← X; R ← X - S;
  if R ≤ -D then Y ← S - D fi;
  if R ≥ D then Y ← S + D fi
done
  
```

X: input signal
 Y: output signal
 S: last output
 R: delta $Y - S$
 D: max. allowed for $|R|$

Iterations in the interval domain (without widening):

| | | | | |
|-----------|----------------|----------------|-----|----------------------|
| $x^{\#0}$ | $x^{\#1}$ | $x^{\#2}$ | ... | $x^{\#n}$ |
| $Y = 0$ | $ Y \leq 144$ | $ Y \leq 160$ | ... | $ Y \leq 128 + 16n$ |

In fact, $Y \in [-128, 128]$ always holds.

To prove that, e.g. $Y \geq -128$, we must be able to:

- **represent** the properties $R = X - S$ and $R \leq -D$
- **combine** them to deduce $S - X \geq D$, and then $Y = S - D \geq X$

The need for relational loop invariants

To prove some invariant after the **end of a loop**,
we often need to find a **loop invariant** of a **more complex form**

relational loop invariant

```
X ← 0; I ← 1;
while • I < 5000 do
  if [0,1] = 1 then X ← X + 1 else X ← X - 1 fi;
  I ← I + 1
done ♦
```

A non-relational analysis finds at ♦ that $I = 5000$ and $X \in \mathbb{Z}$

The best invariant is: $(I = 5000) \wedge (X \in [-4999, 4999]) \wedge (X \equiv 0 [2])$

To find this **non-relational** invariant, we must find a **relational** loop invariant at •: $(-I < X < I) \wedge (X + I \equiv 1 [2]) \wedge (I \in [1, 5000])$,
and apply the loop exit condition $C^\sharp \llbracket I \geq 5000 \rrbracket$

Modular analysis

store the maximum of $X, Y, 0$ into Z

max(X, Y, Z)

```
Z ← X ;  
if Y > Z then Z ← Y ;  
if Z < 0 then Z ← 0;
```

Modular analysis:

- analyze a procedure **once** (procedure summary)
- **reuse** the summary at each call site (instantiation)
⇒ improved efficiency

Modular analysis

store the maximum of $X, Y, 0$ into Z'

max(X, Y, Z)

$X' \leftarrow X; Y' \leftarrow Y; Z' \leftarrow Z;$

$Z' \leftarrow X';$

if $Y' > Z'$ then $Z' \leftarrow Y';$

if $Z' < 0$ then $Z' \leftarrow 0;$

$(Z' \geq X \wedge Z' \geq Y \wedge Z' \geq 0 \wedge X' = X \wedge Y' = Y)$

Modular analysis:

- analyze a procedure **once** (procedure summary)
- reuse** the summary at each call site (instantiation)
 \implies improved efficiency
- infer a **relation** between input X, Y, Z and output X', Y', Z' values, in $\mathcal{P}((\mathbb{V} \rightarrow \mathbb{R}) \times (\mathbb{V} \rightarrow \mathbb{R})) \equiv \mathcal{P}((\mathbb{V} \times \mathbb{V}) \rightarrow \mathbb{R})$
- requires inferring **relational information**

[Anco10], [Jean09]

Linear equality domain

The affine equality domain

Here $\mathbb{I} \in \{\mathbb{Q}, \mathbb{R}\}$.

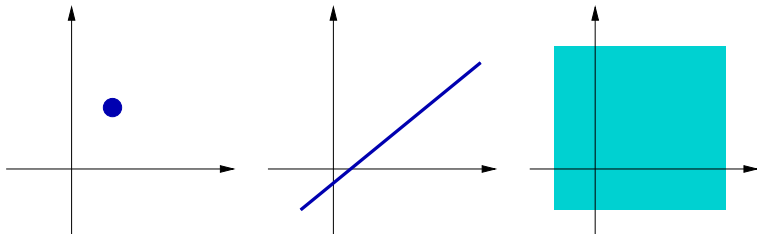
We look for invariants of the form:

$$\bigwedge_j (\sum_{i=1}^n \alpha_{ij} V_i = \beta_j), \alpha_{ij}, \beta_j \in \mathbb{I}$$

where all the α_{ij} and β_j are inferred automatically.

We use a domain of affine spaces proposed by [Karr76]:

$$\mathcal{D}^\# \stackrel{\text{def}}{=} \{ \text{affine subspaces of } \mathbb{V} \rightarrow \mathbb{I} \}$$



Affine equality representation

Machine representation: an affine subspace is represented as

- either the constant \perp^\sharp ,
- or a pair $\langle \mathbf{M}, \vec{C} \rangle$ where
 - $\mathbf{M} \in \mathbb{I}^{m \times n}$ is a $m \times n$ matrix, $n = |\mathbb{V}|$ and $m \leq n$,
 - $\vec{C} \in \mathbb{I}^m$ is a row-vector with m rows.

$\langle \mathbf{M}, \vec{C} \rangle$ represents an equation system, with solutions:

$$\gamma(\langle \mathbf{M}, \vec{C} \rangle) \stackrel{\text{def}}{=} \{ \vec{V} \in \mathbb{I}^n \mid \mathbf{M} \times \vec{V} = \vec{C} \}$$

\mathbf{M} should be in **row echelon form**:

- $\forall i \leq m: \exists k_i: M_{ik_i} = 1$ and
 $\forall c < k_i: M_{ic} = 0, \forall l \neq i: M_{lk_i} = 0,$
- if $i < i'$ then $k_i < k_{i'}$ (*leading index*)

example:

$$\begin{bmatrix} 1 & 0 & 0 & 5 & 0 \\ 0 & 1 & 0 & 6 & 0 \\ 0 & 0 & 1 & 7 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Remarks:

the representation is unique

as $m \leq n = |\mathbb{V}|$, the memory cost is in $\mathcal{O}(n^2)$ at worst

\top is represented as the empty equation system: $m = 0$

Galois connection

Galois connection:

(actually, a Galois insertion)

between arbitrary subsets and affine subsets

$$(\mathcal{P}(\mathbb{I}^n), \subseteq) \begin{matrix} \xleftarrow{\gamma} \\ \xrightarrow{\alpha} \end{matrix} (\text{Aff}(\mathbb{I}^n), \subseteq)$$

- $\gamma(X) \stackrel{\text{def}}{=} X$ (identity)
- $\alpha(X) \stackrel{\text{def}}{=} \text{smallest affine subset containing } X$

$\text{Aff}(\mathbb{I}^n)$ is closed under arbitrary intersections, so we have:

$$\alpha(X) = \bigcap \{ Y \in \text{Aff}(\mathbb{I}^n) \mid X \subseteq Y \}$$

$\text{Aff}(\mathbb{I}^n)$ contains every point in \mathbb{I}^n

we can also construct $\alpha(X)$ by abstract union:

$$\alpha(X) = \bigcup^{\#} \{ \{x\} \mid x \in X \}$$

Notes:

- we have assimilated $\mathbb{V} \rightarrow \mathbb{I}$ to \mathbb{I}^n
- we have used $\text{Aff}(\mathbb{I}^n)$ instead of the matrix representation $\mathcal{D}^{\#}$ for simplicity; a Galois connection also exists between $\mathcal{P}(\mathbb{I}^n)$ and $\mathcal{D}^{\#}$

Normalisation and emptiness testing

Let $\mathbf{M} \times \vec{V} = \vec{C}$ be a system, not necessarily in normal form.

The **Gaussian reduction** *Gauss*($\langle \mathbf{M}, \vec{C} \rangle$) tells in $\mathcal{O}(n^3)$ time:

- whether the system is satisfiable, and in that case
- gives an equivalent system $\langle \mathbf{M}', \vec{C}' \rangle$ in normal form

i.e. returns an element in \mathcal{D}^\sharp .

Principle: reorder lines, and combine lines linearly to eliminate variables

Example:

$$\left\{ \begin{array}{rclcl} 2X & + & Y & + & Z & = & 19 \\ 2X & + & Y & - & Z & = & 9 \\ & & & & 3Z & = & 15 \end{array} \right.$$

↓

$$\left\{ \begin{array}{rcl} X & + & 0.5Y & & = & 7 \\ & & & Z & = & 5 \end{array} \right.$$

Affine equality operators

Applications

If $\mathcal{X}^\#, \mathcal{Y}^\# \neq \perp^\#$, we define:

$$\mathcal{X}^\# \cap^\# \mathcal{Y}^\# \stackrel{\text{def}}{=} \text{Gauss} \left(\left\langle \left[\begin{array}{c} \mathbf{M}_{\mathcal{X}^\#} \\ \mathbf{M}_{\mathcal{Y}^\#} \end{array} \right], \left[\begin{array}{c} \vec{C}_{\mathcal{X}^\#} \\ \vec{C}_{\mathcal{Y}^\#} \end{array} \right] \right\rangle \right)$$

$$\mathcal{X}^\# =^\# \mathcal{Y}^\# \stackrel{\text{def}}{\iff} \mathbf{M}_{\mathcal{X}^\#} = \mathbf{M}_{\mathcal{Y}^\#} \quad \text{and} \quad \vec{C}_{\mathcal{X}^\#} = \vec{C}_{\mathcal{Y}^\#}$$

$$\mathcal{X}^\# \subseteq^\# \mathcal{Y}^\# \stackrel{\text{def}}{\iff} \mathcal{X}^\# \cap^\# \mathcal{Y}^\# =^\# \mathcal{X}^\#$$

$$\mathbf{C}^\#[\sum_j \alpha_j V_j - \beta = 0] \mathcal{X}^\# \stackrel{\text{def}}{=} \text{Gauss} \left(\left\langle \left[\begin{array}{c} \mathbf{M}_{\mathcal{X}^\#} \\ \alpha_1 \cdots \alpha_n \end{array} \right], \left[\begin{array}{c} \vec{C}_{\mathcal{X}^\#} \\ \beta \end{array} \right] \right\rangle \right)$$

$$\mathbf{C}^\#[e \bowtie 0] \mathcal{X}^\# \stackrel{\text{def}}{=} \mathcal{X}^\# \quad \text{for other tests}$$

Remark:

$\subseteq^\#, =^\#, \cap^\#, =^\#$ and $\mathbf{C}^\#[\sum_j \alpha_j V_j - \beta = 0]$ are **exact**:

$$\mathcal{X}^\# \subseteq^\# \mathcal{Y}^\# \iff \gamma(\mathcal{X}^\#) \subseteq \gamma(\mathcal{Y}^\#), \quad \gamma(\mathcal{X}^\# \cap^\# \mathcal{Y}^\#) = \gamma(\mathcal{X}^\#) \cap \gamma(\mathcal{Y}^\#), \dots$$

Generator representation

Generator representation

An affine subspace can also be represented as a set of **vector generators** $\vec{G}_1, \dots, \vec{G}_m$ and an **origin point** \vec{O} , denoted as $[\mathbf{G}, \vec{O}]$.

$$\gamma([\mathbf{G}, \vec{O}]) \stackrel{\text{def}}{=} \{ \mathbf{G} \times \vec{\lambda} + \vec{O} \mid \vec{\lambda} \in \mathbb{I}^m \} \quad (\mathbf{G} \in \mathbb{I}^{n \times m}, \vec{O} \in \mathbb{I}^n)$$

We can **switch** between a generator and a constraint representation:

- From generators to constraints: $\langle \mathbf{M}, \vec{C} \rangle = \text{Cons}([\mathbf{G}, \vec{O}])$

Write the system $\vec{V} = \mathbf{G} \times \vec{\lambda} + \vec{O}$ with variables $\vec{V}, \vec{\lambda}$.

Solve it in $\vec{\lambda}$ (by row operations).

Keep the constraints involving only \vec{V} .

$$\text{e.g. } \begin{cases} X = \lambda + 2 \\ Y = 2\lambda + \mu + 3 \\ Z = \mu \end{cases} \implies \begin{cases} X - 2 = \lambda \\ -2X + Y + 1 = \mu \\ 2X - Y + Z - 1 = 0 \end{cases}$$

The result is: $2X - Y + Z = 1$.

Generator representation (cont.)

- From constraints to generators: $[\mathbf{G}, \vec{O}] \stackrel{\text{def}}{=} \text{Gen}(\langle \mathbf{M}, \vec{C} \rangle)$

Assume $\langle \mathbf{M}, \vec{C} \rangle$ is normalized.

For each non-leading variable V , assign a distinct λ_V ,
solve leading variables in terms of non-leading ones.

$$\text{e.g. } \begin{cases} X + 0.5Y & = & 7 \\ Z & = & 5 \end{cases} \implies \begin{bmatrix} -0.5 \\ 1 \\ 0 \end{bmatrix} \lambda_Y + \begin{bmatrix} 7 \\ 0 \\ 5 \end{bmatrix}$$

Affine equality operators (cont.)

Applications

Given $\mathcal{X}^\#, \mathcal{Y}^\# \neq \perp^\#$, we define:

$$\mathcal{X}^\# \cup^\# \mathcal{Y}^\# \stackrel{\text{def}}{=} \text{Cons} \left(\left[\mathbf{G}_{\mathcal{X}^\#} \mathbf{G}_{\mathcal{Y}^\#} (\vec{O}_{\mathcal{Y}^\#} - \vec{O}_{\mathcal{X}^\#}), \vec{O}_{\mathcal{X}^\#} \right] \right)$$

$$\mathbf{C}^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \text{Cons} \left(\left[\mathbf{G}_{\mathcal{X}^\#} \vec{x}_j, \vec{O}_{\mathcal{X}^\#} \right] \right)$$

$$\mathbf{C}^\# \llbracket V_j \leftarrow \sum_i \alpha_i V_i + \beta \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=}$$

if $\alpha_j = 0$, $(\mathbf{C}^\# \llbracket \sum_i \alpha_i V_i - V_j + \beta = 0 \rrbracket \circ \mathbf{C}^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket) \mathcal{X}^\#$

if $\alpha_j \neq 0$, $\mathcal{X}^\#$ where V_j is replaced with $(V_j - \sum_{i \neq j} \alpha_i V_i - \beta) / \alpha_j$

(proofs on next slide)

$$\mathbf{C}^\# \llbracket V_j \leftarrow e \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} \mathbf{C}^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket \mathcal{X}^\# \text{ for other assignments}$$

Remarks:

- $\cup^\#$ is **optimal**, but not exact.
- $\mathbf{C}^\# \llbracket V_j \leftarrow \sum_i \alpha_i V_i + \beta \rrbracket$ and $\mathbf{C}^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket$ are **exact**.

Affine assignments: proofs

$$\mathbb{C}^\# \llbracket V_j \leftarrow \sum_i \alpha_i V_i + \beta \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=}$$

if $\alpha_j = 0$, $(\mathbb{C}^\# \llbracket \sum_i \alpha_i V_i - V_j + \beta = 0 \rrbracket \circ \mathbb{C}^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket) \mathcal{X}^\#$

if $\alpha_j \neq 0$, $\mathcal{X}^\#$ where V_j is replaced with $(V_j - \sum_{i \neq j} \alpha_i V_i - \beta) / \alpha_j$

Proof sketch:

we use the following identities in the concrete

non-invertible assignment: $\alpha_j = 0$

$\mathbb{C} \llbracket V_j \leftarrow e \rrbracket = \mathbb{C} \llbracket V_j \leftarrow e \rrbracket \circ \mathbb{C} \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket$ as the value of V_j is not used in e

so: $\mathbb{C} \llbracket V_j \leftarrow e \rrbracket = \mathbb{C} \llbracket V_j - e = 0 \rrbracket \circ \mathbb{C} \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket$

\implies reduces the assignment to a test

invertible assignment: $\alpha_j \neq 0$

$\mathbb{C} \llbracket V_j \leftarrow e \rrbracket \subsetneq \mathbb{C} \llbracket V_j \leftarrow e \rrbracket \circ \mathbb{C} \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket$ as e depends on V
(e.g., $\mathbb{C} \llbracket V \leftarrow V + 1 \rrbracket \neq \mathbb{C} \llbracket V \leftarrow V + 1 \rrbracket \circ \mathbb{C} \llbracket V \leftarrow [-\infty, +\infty] \rrbracket$)

$$\begin{aligned} \rho \in \mathbb{C} \llbracket V_j \leftarrow e \rrbracket R &\iff \exists \rho' \in R: \rho = \rho' [V_j \mapsto \sum_i \alpha_i \rho'(V_i) + \beta] \\ &\iff \exists \rho' \in R: \rho[V_j \mapsto (\rho(V_j) - \sum_{i \neq j} \alpha_i \rho'(V_i) - \beta) / \alpha_j] = \rho' \\ &\iff \rho[V_j \mapsto (\rho(V_j) - \sum_{i \neq j} \alpha_i \rho(V_i) - \beta) / \alpha_j] \in R \end{aligned}$$

\implies reduces the assignment to a substitution by the inverse expression

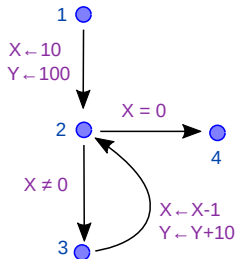
Analysis example

No infinite increasing chain: we can iterate without widening.

Forward analysis example:

```

1X ← 10; Y ← 100;
while 2X ≠ 0 do 3
  X ← X-1;
  Y ← Y+10
done4
  
```



| ℓ | $\mathcal{X}_\ell^{\#0}$ | $\mathcal{X}_\ell^{\#1}$ | $\mathcal{X}_\ell^{\#2}$ | $\mathcal{X}_\ell^{\#3}$ | $\mathcal{X}_\ell^{\#4}$ |
|--------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| 1 | $\top^\#$ | $\top^\#$ | $\top^\#$ | $\top^\#$ | $\top^\#$ |
| 2 | $\perp^\#$ | (10, 100) | (10, 100) | $10X + Y = 200$ | $10X + Y = 200$ |
| 3 | $\perp^\#$ | $\perp^\#$ | (10, 100) | (10, 100) | $10X + Y = 200$ |
| 4 | $\perp^\#$ | $\perp^\#$ | $\perp^\#$ | $\perp^\#$ | (0, 200) |

Note in particular:

$$\mathcal{X}_2^{\#3} = \{(10, 100)\} \cup^\# \{(9, 110)\} = \{(X, Y) \mid 10X + Y = 200\}$$

Backward affine equality operators

Backward assignments:

$$\overleftarrow{C}^{\#} \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket (\mathcal{X}^{\#}, \mathcal{R}^{\#}) \stackrel{\text{def}}{=} \mathcal{X}^{\#} \cap^{\#} (C^{\#} \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket \mathcal{R}^{\#})$$

$$\overleftarrow{C}^{\#} \llbracket V_j \leftarrow \sum_i \alpha_i V_i + \beta \rrbracket (\mathcal{X}^{\#}, \mathcal{R}^{\#}) \stackrel{\text{def}}{=} \mathcal{X}^{\#} \cap^{\#} (\mathcal{R}^{\#} \text{ where } V_j \text{ is replaced with } (\sum_i \alpha_i V_i + \beta))$$

(reduces to a substitution by the (non-inverted) expression)

$$\overleftarrow{C}^{\#} \llbracket V_j \leftarrow e \rrbracket (\mathcal{X}^{\#}, \mathcal{R}^{\#}) \stackrel{\text{def}}{=} \overleftarrow{C}^{\#} \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket (\mathcal{X}^{\#}, \mathcal{R}^{\#})$$

for other assignments

Remarks:

- $\overleftarrow{C}^{\#} \llbracket V_j \leftarrow \sum_i \alpha_i V_i + \beta \rrbracket$ and $\overleftarrow{C}^{\#} \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket$ are **exact**

Constraint-only equality domain

In fact [Karr76] does not use the generator representation.

(rationale: few constraints but many generators in practice)

We need to redefine two operators: forgetting and union.

- $C^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket$

Idea:

We have to remove all the occurrences of V_j
but reduce the number of equations by only one

Algorithm:

Pick the row $\langle \vec{M}_i, C_i \rangle$ such that $M_{ij} \neq 0$ and i maximal.

Use it to eliminate all non-0 occurrences of V_j in \mathbf{M} .

(i maximal $\implies \mathbf{M}$ stays in row echelon form)

Then remove the row $\langle \vec{M}_i, C_i \rangle$.

e.g. forgetting Z : $\begin{cases} X + Z = 10 \\ Y + Z = 7 \end{cases} \implies \begin{cases} X - Y = 3 \end{cases}$

The operator is exact.

Constraint-only equality domain (cont.)

- $\langle \mathbf{M}, \vec{C} \rangle \cup^\# \langle \mathbf{N}, \vec{D} \rangle$

Idea: unify columns 1 to n in $\langle \mathbf{M}, \vec{C} \rangle$ and $\langle \mathbf{N}, \vec{D} \rangle$ using row operations.

Algorithm sketch:

Assume that we have unified columns 1 to k to get $\begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix}$, arguments are in row echelon form, and we have to unify at column $k + 1$: ${}^t(\vec{0} \ 1 \ \vec{0})$ with ${}^t(\vec{\beta} \ 0 \ \vec{0})$

$$\begin{pmatrix} \mathbf{R} & \vec{0} & \mathbf{M}_1 \\ \vec{0} & 1 & \vec{M}_2 \\ \mathbf{0} & \vec{0} & \mathbf{M}_3 \end{pmatrix}, \begin{pmatrix} \mathbf{R} & \vec{\beta} & \mathbf{N}_1 \\ \vec{0} & 0 & \vec{N}_2 \\ \mathbf{0} & \vec{0} & \mathbf{N}_3 \end{pmatrix} \implies \begin{pmatrix} \mathbf{R} & \vec{\beta} & \mathbf{M}'_1 \\ \vec{0} & 0 & \vec{0} \\ \mathbf{0} & \vec{0} & \mathbf{M}_3 \end{pmatrix}, \begin{pmatrix} \mathbf{R} & \vec{\beta} & \mathbf{N}_1 \\ \vec{0} & 0 & \vec{N}_2 \\ \mathbf{0} & \vec{0} & \mathbf{N}_3 \end{pmatrix}$$

Use the row $(\vec{0} \ 1 \ \vec{M}_2)$ to create $\vec{\beta}$ in the left argument

Then remove the row $(\vec{0} \ 1 \ \vec{M}_2)$

The right argument is unchanged

\implies we have now unified columns 1 to $k + 1$

Unifying ${}^t(\vec{\alpha} \ 0 \ \vec{0})$ and ${}^t(\vec{0} \ 1 \ \vec{0})$ is similar

Unifying ${}^t(\vec{\alpha} \ 0 \ \vec{0})$ and ${}^t(\vec{\beta} \ 0 \ \vec{0})$ is a bit more complicated... see [Karr76]

No other case possible as we are in row echelon form

A note on integers

Suppose now that $\mathbb{I} = \mathbb{Z}$.

- \mathbb{Z} is not closed under affine operations: $(x/y) \times y \neq x$,
- Gaussian reduction implemented in \mathbb{Z} is **unsound**.
(e.g. unsound normalization $2X + Y = 19 \not\Rightarrow X = 9$, by truncation)

One possible solution:

- keep a representation using matrices with coefficients in \mathbb{Q} ,
- keep all abstract operators as in \mathbb{Q} ,
- change the concretization into: $\gamma_{\mathbb{Z}}(\mathcal{X}^{\#}) \stackrel{\text{def}}{=} \gamma(\mathcal{X}^{\#}) \cap \mathbb{Z}^n$.

With respect to $\gamma_{\mathbb{Z}}$, the operators are **no longer best / exact**.

Example: where $\mathcal{X}^{\#}$ is the equation $Y = 2X$

- $\gamma_{\mathbb{Z}}(\mathcal{X}^{\#}) = \{ (X, Y) \mid X \in \mathbb{Z}, Y = 2X \}$
- $(C[X \leftarrow 0] \circ \gamma_{\mathbb{Z}})\mathcal{X}^{\#} = \{ (X, Y) \mid X = 0, Y \text{ is even} \}$
- $(\gamma_{\mathbb{Z}} \circ C^{\#}[X \leftarrow 0])\mathcal{X}^{\#} = \{ (X, Y) \mid X = 0, Y \in \mathbb{Z} \}$

\Rightarrow The analysis forgets the “intergrerness” of variables.

The congruence equality domain

Another possible solution: use a more expressive domain.

We look for invariants of the form: $\bigwedge_j \left(\sum_{i=1}^n m_{ij} V_i \equiv c_j [k_j] \right)$.

Algorithms:

- there exists minimal forms (but not unique), computed using an extension of **Euclide's algorithm**,
- there is a dual representation: $\{ \mathbf{G} \times \vec{\lambda} + \vec{O} \mid \vec{\lambda} \in \mathbb{Z}^m \}$, and passage algorithms,
- see [Gran91].

Polyhedron domain

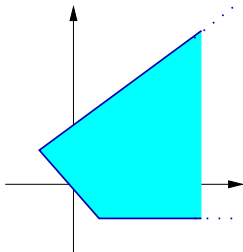
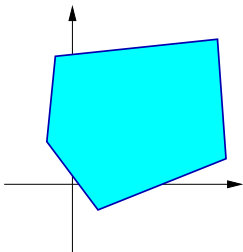
The polyhedron domain

Here again, $\mathbb{I} \in \{\mathbb{Q}, \mathbb{R}\}$.

We look for invariants of the form: $\bigwedge_j \left(\sum_{i=1}^n \alpha_{ij} v_i \geq \beta_j \right)$.

We use the polyhedron domain proposed by [Cous78]:

$$\mathcal{D}^\# \stackrel{\text{def}}{=} \{\text{closed convex polyhedra of } \mathbb{V} \rightarrow \mathbb{I}\}$$



Note: polyhedra need not be bounded (\neq polytopes).

Double description of polyhedra

Polyhedra have **dual** representations (Weyl–Minkowski Theorem).
(see [Schr86])

Constraint representation

$\langle \mathbf{M}, \vec{C} \rangle$ with $\mathbf{M} \in \mathbb{I}^{m \times n}$ and $\vec{C} \in \mathbb{I}^m$
represents: $\gamma(\langle \mathbf{M}, \vec{C} \rangle) \stackrel{\text{def}}{=} \{ \vec{V} \mid \mathbf{M} \times \vec{V} \geq \vec{C} \}$

We will also often use a **constraint set notation** $\{ \sum_i \alpha_{ij} V_i \geq \beta_j \}$.

Generator representation

$[\mathbf{P}, \mathbf{R}]$ where

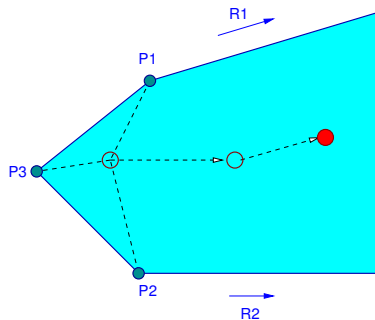
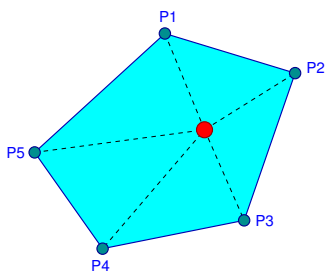
- $\mathbf{P} \in \mathbb{I}^{n \times p}$ is a set of p **points**: $\vec{P}_1, \dots, \vec{P}_p$
- $\mathbf{R} \in \mathbb{I}^{n \times r}$ is a set of r **rays**: $\vec{R}_1, \dots, \vec{R}_r$

$\gamma([\mathbf{P}, \mathbf{R}]) \stackrel{\text{def}}{=} \left\{ \left(\sum_{j=1}^p \alpha_j \vec{P}_j \right) + \left(\sum_{j=1}^r \beta_j \vec{R}_j \right) \mid \forall j, \alpha_j, \beta_j \geq 0, \sum_{j=1}^p \alpha_j = 1 \right\}$

Double description of polyhedra (cont.)

Generator representation examples:

$$\gamma([\mathbf{P}, \mathbf{R}]) \stackrel{\text{def}}{=} \{ (\sum_{j=1}^p \alpha_j \vec{P}_j) + (\sum_{j=1}^r \beta_j \vec{R}_j) \mid \forall j, \alpha_j, \beta_j \geq 0 : \sum_{j=1}^p \alpha_j = 1 \}$$



- the points define a bounded convex hull
- the rays allow unbounded polyhedra

Origin of duality

$$\text{Dual } A^* \stackrel{\text{def}}{=} \{ \vec{x} \in \mathbb{I}^n \mid \forall \vec{a} \in A, \vec{a} \cdot \vec{x} \leq 0 \}$$

- $\{\vec{a}\}^*$ and $\{\lambda \vec{r} \mid \lambda \geq 0\}^*$ are half-spaces,
- $(A \cup B)^* = A^* \cap B^*$,
- if A is convex, closed, and $\vec{0} \in A$, then $A^{**} = A$.

Duality on polyhedral cones:

Cone: $C = \{ \vec{V} \mid \mathbf{M} \times \vec{V} \geq \vec{0} \}$ or $C = \{ \sum_{j=1}^r \beta_j \vec{R}_j \mid \forall j, \beta_j \geq 0 \}$
 (polyhedron with no vertex, except $\vec{0}$)

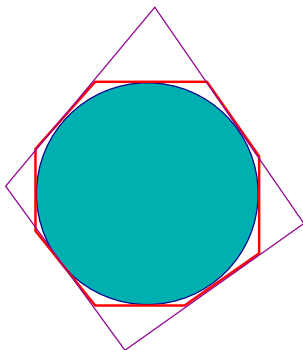
- C^* is also a polyhedral cone,
- $C^{**} = C$,
- a **ray of C** corresponds to a **constraint of C^*** ,
- a **constraint of C** corresponds to a **ray of C^*** .

Extension to polyhedra: by homogenisation to polyhedral cones:

$$C(P) \stackrel{\text{def}}{=} \{ \lambda \vec{V} \mid \lambda \geq 0, (V_1, \dots, V_n) \in \gamma(P), V_{n+1} = 1 \} \subseteq \mathbb{I}^{n+1}$$

(polyhedron in $\mathbb{I}^n \simeq$ polyhedral cone in \mathbb{I}^{n+1})

Polyhedra representations



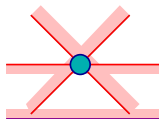
- **No best abstraction** α
(e.g., a disc has infinitely many polyhedral over-approximations, but no best one)
- **No memory bound** on the representations

Polyhedra representations

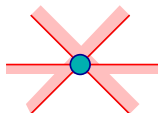
Minimal representations

- A constraint / generator system is **minimal** if no constraint / generator can be omitted without changing the concretization
- Minimal representations are **not unique**
- No memory bound even on minimal representations

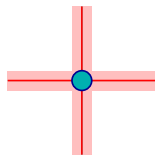
Example: three different constraint representations for a point



(a)



(b)



(c)

- (a) $y + x \geq 0, y - x \geq 0, y \leq 0, y \geq -5$
- (b) $y + x \geq 0, y - x \geq 0, y \leq 0$
- (c) $x \leq 0, x \geq 0, y \leq 0, y \geq 0$

(non minimal)

(minimal)

(minimal)

Chernikova's algorithm

Algorithm by [Cher68], improved by [LeVe92] to switch from a constraint system to an equivalent generator system

Why? most operators are easier on one representation

Notes:

- By **duality**, we can use the same algorithm to switch from generators to constraints
- The minimal generator system can be **exponential** in the original constraint system
(e.g., hypercube: $2n$ constraints, 2^n vertices)
- **Equality** constraints and **lines** (pairs of opposed rays) may be handled separately and more efficiently

Chernikova's algorithm (cont.)

Algorithm: incrementally add constraints one by one

Start with: $\begin{cases} \mathbf{P}_0 = \{ (0, \dots, 0) \} & \text{(origin)} \\ \mathbf{R}_0 = \{ \vec{x}_i, -\vec{x}_i \mid 1 \leq i \leq n \} & \text{(axes)} \end{cases}$

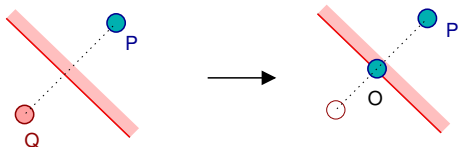
For each constraint $\vec{M}_k \cdot \vec{V} \geq C_k \in \langle \mathbf{M}, \vec{C} \rangle$, update $[\mathbf{P}_{k-1}, \mathbf{R}_{k-1}]$ to $[\mathbf{P}_k, \mathbf{R}_k]$.

Start with $\mathbf{P}_k = \mathbf{R}_k = \emptyset$,

- for any $\vec{P} \in \mathbf{P}_{k-1}$ s.t. $\vec{M}_k \cdot \vec{P} \geq C_k$, add \vec{P} to \mathbf{P}_k
- for any $\vec{R} \in \mathbf{R}_{k-1}$ s.t. $\vec{M}_k \cdot \vec{R} \geq 0$, add \vec{R} to \mathbf{R}_k
- for any $\vec{P}, \vec{Q} \in \mathbf{P}_{k-1}$ s.t. $\vec{M}_k \cdot \vec{P} > C_k$ and $\vec{M}_k \cdot \vec{Q} < C_k$, add to \mathbf{P}_k :

$$\vec{O} \stackrel{\text{def}}{=} \frac{C_k - \vec{M}_k \cdot \vec{Q}}{\vec{M}_k \cdot \vec{P} - \vec{M}_k \cdot \vec{Q}} \vec{P} - \frac{C_k - \vec{M}_k \cdot \vec{P}}{\vec{M}_k \cdot \vec{P} - \vec{M}_k \cdot \vec{Q}} \vec{Q}$$

i.e., move Q towards P along $[Q, P]$ until it saturates the constraint

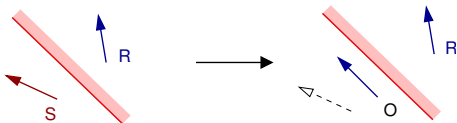


Chernikova's algorithm (cont.)

- for any $\vec{R}, \vec{S} \in \mathbf{R}_{k-1}$ s.t. $\vec{M}_k \cdot \vec{R} > 0$ and $\vec{M}_k \cdot \vec{S} < 0$, add to \mathbf{R}_k :

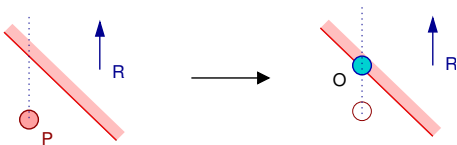
$$\vec{O} \stackrel{\text{def}}{=} (\vec{M}_k \cdot \vec{S})\vec{R} - (\vec{M}_k \cdot \vec{R})\vec{S}$$

i.e., rotate S towards R until it is parallel to the constraint



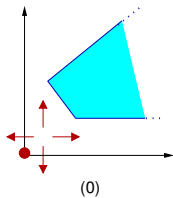
- for any $\vec{P} \in \mathbf{P}_{k-1}, \vec{R} \in \mathbf{R}_{k-1}$ s.t.
either $\vec{M}_k \cdot \vec{P} > C_k$ and $\vec{M}_k \cdot \vec{R} < 0$, or $\vec{M}_k \cdot \vec{P} < C_k$ and $\vec{M}_k \cdot \vec{R} > 0$

$$\text{add to } \mathbf{P}_k: \vec{O} \stackrel{\text{def}}{=} \vec{P} + \frac{C_k - \vec{M}_k \cdot \vec{P}}{\vec{M}_k \cdot \vec{R}} \vec{R}$$



Chernikova's algorithm example

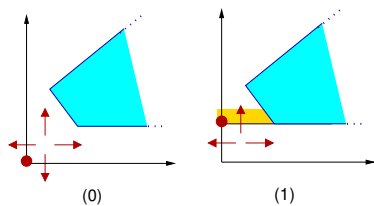
Example:



$$\mathbf{P}_0 = \{(0, 0)\}$$

$$\mathbf{R}_0 = \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$$

Chernikova's algorithm example

Example:

$$Y \geq 1$$

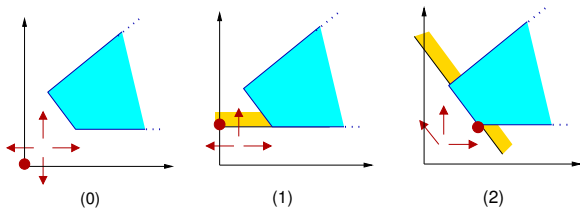
$$P_0 = \{(0, 0)\}$$

$$P_1 = \{(0, 1)\}$$

$$R_0 = \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$$

$$R_1 = \{(1, 0), (-1, 0), (0, 1)\}$$

Chernikova's algorithm example

Example:

$$Y \geq 1$$

$$X + Y \geq 3$$

$$\mathbf{P}_0 = \{(0, 0)\}$$

$$\mathbf{P}_1 = \{(0, 1)\}$$

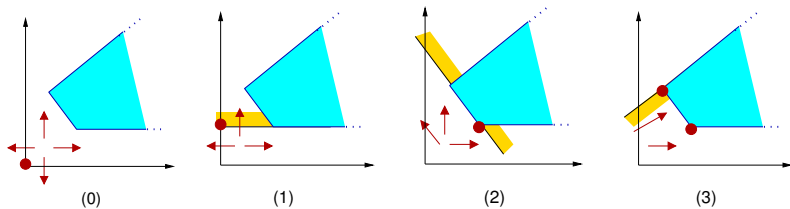
$$\mathbf{P}_2 = \{(2, 1)\}$$

$$\mathbf{R}_0 = \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$$

$$\mathbf{R}_1 = \{(1, 0), (-1, 0), (0, 1)\}$$

$$\mathbf{R}_2 = \{(1, 0), (-1, 1), (0, 1)\}$$

Chernikova's algorithm example

Example:

| | | |
|----------------|----------------------------|--|
| | $P_0 = \{(0, 0)\}$ | $R_0 = \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$ |
| $Y \geq 1$ | $P_1 = \{(0, 1)\}$ | $R_1 = \{(1, 0), (-1, 0), (0, 1)\}$ |
| $X + Y \geq 3$ | $P_2 = \{(2, 1)\}$ | $R_2 = \{(1, 0), (-1, 1), (0, 1)\}$ |
| $X - Y \leq 1$ | $P_3 = \{(2, 1), (1, 2)\}$ | $R_3 = \{(0, 1), (1, 1)\}$ |

Redundancy removal

Goal: only introduce non-redundant points and rays during Chernikova's algorithm

Definitions (for rays in polyhedral cones)

Given $C = \{ \vec{V} \mid \mathbf{M} \times \vec{V} \geq \vec{0} \} = \{ \mathbf{R} \times \vec{\beta} \mid \vec{\beta} \geq \vec{0} \}$.

- \vec{R} saturates $\vec{M}_k \cdot \vec{V} \geq 0 \stackrel{\text{def}}{\iff} \vec{M}_k \cdot \vec{R} = 0$
- $S(\vec{R}, C) \stackrel{\text{def}}{=} \{ k \mid \vec{M}_k \cdot \vec{R} = 0 \}$.

Theorem:

assume C has no line ($\exists \vec{L} \neq \vec{0}$ s.t. $\forall \alpha, \alpha \vec{L} \in C$)

\vec{R} is non-redundant w.r.t. $\mathbf{R} \iff \exists \vec{R}_i \in \mathbf{R}, S(\vec{R}, C) \subseteq S(\vec{R}_i, C)$

- $S(\vec{R}_i, C), \vec{R}_i \in \mathbf{R}$ is maintained during Chernikova's algorithm in a **saturation matrix**
- extension possible to polyhedra and lines
- various improvements exist [LeVe92]

Operators on polyhedra

Given $\mathcal{X}^\#, \mathcal{Y}^\# \neq \perp^\#$, we define:

$$\mathcal{X}^\# \subseteq^\# \mathcal{Y}^\# \stackrel{\text{def}}{\iff} \begin{cases} \forall \vec{P} \in \mathbf{P}_{\mathcal{X}^\#}, \mathbf{M}_{\mathcal{Y}^\#} \times \vec{P} \geq \vec{C}_{\mathcal{Y}^\#} \\ \forall \vec{R} \in \mathbf{R}_{\mathcal{X}^\#}, \mathbf{M}_{\mathcal{Y}^\#} \times \vec{R} \geq \vec{0} \end{cases}$$

(every generator of $\mathcal{X}^\#$ must satisfy every constraint in $\mathcal{Y}^\#$)

$$\mathcal{X}^\# =^\# \mathcal{Y}^\# \stackrel{\text{def}}{\iff} \mathcal{X}^\# \subseteq^\# \mathcal{Y}^\# \text{ and } \mathcal{Y}^\# \subseteq^\# \mathcal{X}^\#$$

$$\mathcal{X}^\# \cap^\# \mathcal{Y}^\# \stackrel{\text{def}}{=} \left\langle \left[\begin{array}{c} \mathbf{M}_{\mathcal{X}^\#} \\ \mathbf{M}_{\mathcal{Y}^\#} \end{array} \right], \left[\begin{array}{c} \vec{C}_{\mathcal{X}^\#} \\ \vec{C}_{\mathcal{Y}^\#} \end{array} \right] \right\rangle$$

(set union of sets of constraints)

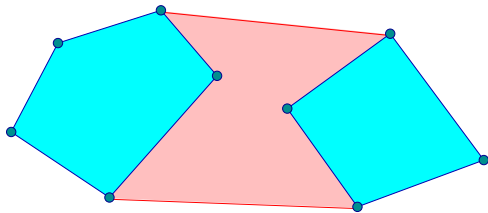
Remarks:

- $\subseteq^\#, =^\#$ and $\cap^\#$ are **exact**.

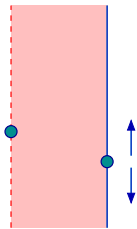
Operators on polyhedra: join

Join: $\mathcal{X}^\# \cup^\# \mathcal{Y}^\# \stackrel{\text{def}}{=} [[\mathbf{P}_{\mathcal{X}^\#} \ \mathbf{P}_{\mathcal{Y}^\#}], [\mathbf{R}_{\mathcal{X}^\#} \ \mathbf{R}_{\mathcal{Y}^\#}]]$ (join generator sets)

Examples:



two polytopes



a point and a line

$\cup^\#$ is **optimal**:

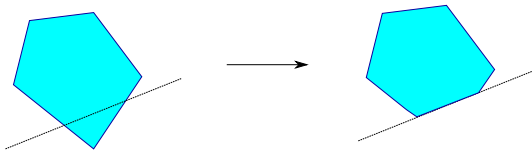
we get the **topological closure of the convex hull** of $\gamma(\mathcal{X}^\#) \cup \gamma(\mathcal{Y}^\#)$

Operators on polyhedra (cont.)

Forward operators: affine tests

$$C^\#[\sum_i \alpha_i V_i + \beta \geq 0] \mathcal{X}^\# \stackrel{\text{def}}{=} \left\langle \begin{bmatrix} \mathbf{M} \mathcal{X}^\# \\ \alpha_1 \cdots \alpha_n \end{bmatrix}, \begin{bmatrix} \vec{C} \mathcal{X}^\# \\ -\beta \end{bmatrix} \right\rangle$$

$$C^\#[\sum_i \alpha_i V_i + \beta = 0] \mathcal{X}^\# \stackrel{\text{def}}{=} (C^\#[\sum_i \alpha_i V_i + \beta \geq 0] \circ C^\#[\sum_i (-\alpha_i) V_i - \beta \geq 0]) \mathcal{X}^\#$$

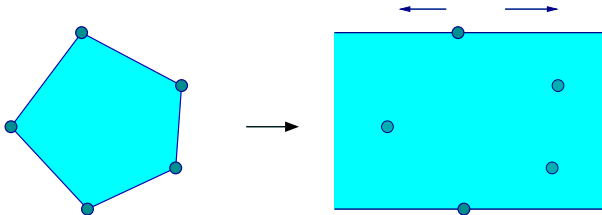


These test operators are exact.

Operators on polyhedra (cont.)

Forward operators: forget

$$C^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=} [\mathbf{P}_{\mathcal{X}^\#}, [\mathbf{R}_{\mathcal{X}^\#} \ \bar{x}_j \ (-\bar{x}_j)]]$$



This operator is exact.

It is also a sound abstraction for any assignment.

Operators on polyhedra (cont.)

Forward operators: affine assignments

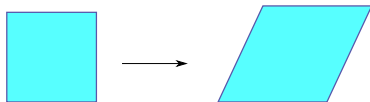
$$C^\# \llbracket V_j \leftarrow \sum_i \alpha_i V_i + \beta \rrbracket \mathcal{X}^\# \stackrel{\text{def}}{=}$$

if $\alpha_j = 0$, $(C^\# \llbracket \sum_i \alpha_i V_i - V_j + \beta = 0 \rrbracket \circ C^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket) \mathcal{X}^\#$

if $\alpha_j \neq 0$, $\langle \mathbf{M}, \vec{C} \rangle$ where V_j is replaced with $\frac{1}{\alpha_j} (V_j - \sum_{i \neq j} \alpha_i V_i - \beta)$

Examples :

$$X \leftarrow X + Y$$



$$X \leftarrow Y$$



Affine assignments are exact.

They could also be defined on generator systems.

Operators on polyhedra (cont.)

Backward assignments:

$$\overleftarrow{C}^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket (\mathcal{X}^\#, \mathcal{R}^\#) \stackrel{\text{def}}{=} \mathcal{X}^\# \cap^\# (C^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket \mathcal{R}^\#)$$

$$\overleftarrow{C}^\# \llbracket V_j \leftarrow \sum_i \alpha_i V_i + \beta \rrbracket (\mathcal{X}^\#, \mathcal{R}^\#) \stackrel{\text{def}}{=} \mathcal{X}^\# \cap^\# (\mathcal{R}^\# \text{ where } V_j \text{ is replaced with } (\sum_i \alpha_i V_i + \beta))$$

$$\overleftarrow{C}^\# \llbracket V_j \leftarrow e \rrbracket (\mathcal{X}^\#, \mathcal{R}^\#) \stackrel{\text{def}}{=} \overleftarrow{C}^\# \llbracket V_j \leftarrow [-\infty, +\infty] \rrbracket (\mathcal{X}^\#, \mathcal{R}^\#)$$

for other assignments

Note: identical to the case of linear equalities.

Polyhedra widening

$\mathcal{D}^\#$ has strictly increasing infinite chains \implies we need a widening

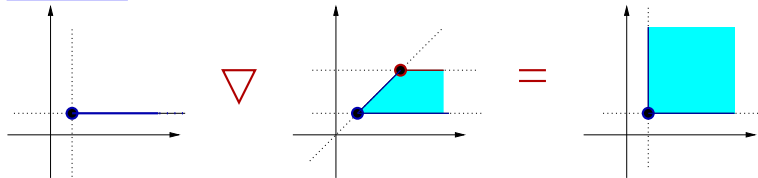
Definition:

Take $\mathcal{X}^\#$ and $\mathcal{Y}^\#$ in minimal constraint-set form

$$\mathcal{X}^\# \nabla \mathcal{Y}^\# \stackrel{\text{def}}{=} \{c \in \mathcal{X}^\# \mid \mathcal{Y}^\# \subseteq^\# \{c\}\}$$

We suppress any unstable constraint $c \in \mathcal{X}^\#$, i.e., $\mathcal{Y}^\# \not\subseteq^\# \{c\}$

Example:



Polyhedra widening

$\mathcal{D}^\#$ has strictly increasing infinite chains \implies we need a widening

Definition:

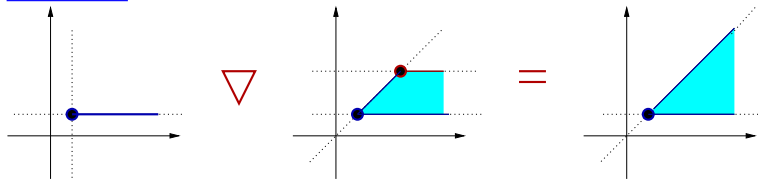
Take $\mathcal{X}^\#$ and $\mathcal{Y}^\#$ in minimal constraint-set form

$$\mathcal{X}^\# \nabla \mathcal{Y}^\# \stackrel{\text{def}}{=} \begin{aligned} & \{c \in \mathcal{X}^\# \mid \mathcal{Y}^\# \subseteq^\# \{c\}\} \\ \cup & \{c \in \mathcal{Y}^\# \mid \exists c' \in \mathcal{X}^\# : \mathcal{X}^\# =^\# (\mathcal{X}^\# \setminus c') \cup \{c\}\} \end{aligned}$$

We suppress any unstable constraint $c \in \mathcal{X}^\#$, i.e., $\mathcal{Y}^\# \not\subseteq^\# \{c\}$

We also keep constraints $c \in \mathcal{Y}^\#$ equivalent to those in $\mathcal{X}^\#$, i.e., when $\exists c' \in \mathcal{X}^\# : \mathcal{X}^\# =^\# (\mathcal{X}^\# \setminus c') \cup \{c\}$

Example:



Example analysis

Example program

```

X ← 2; I ← 0;
while • I < 10 do
  if [0,1] = 0 then X ← X + 2 else X ← X - 3 fi;
  I ← I + 1
done ♦

```

Loop invariant:

Increasing iterations with widening at • give:

$$\begin{aligned}
 \mathcal{X}_1^\# &= \{X = 2, I = 0\} \\
 \mathcal{X}_2^\# &= \{X = 2, I = 0\} \nabla (\{X = 2, I = 0\} \cup^\# \{X \in [-1, 4], I = 1\}) \\
 &= \{X = 2, I = 0\} \nabla \{I \in [0, 1], 2 - 3I \leq X \leq 2I + 2\} \\
 &= \{I \geq 0, 2 - 3I \leq X \leq 2I + 2\}
 \end{aligned}$$

Decreasing iterations (to find $I \leq 10$):

$$\begin{aligned}
 \mathcal{X}_3^\# &= \{X = 2, I = 0\} \cup^\# \{I \in [1, 10], 2 - 3I \leq X \leq 2I + 2\} \\
 &= \{I \in [0, 10], 2 - 3I \leq X \leq 2I + 2\}
 \end{aligned}$$

We find, at the end of the loop ♦: $I = 10 \wedge X \in [-28, 22]$.

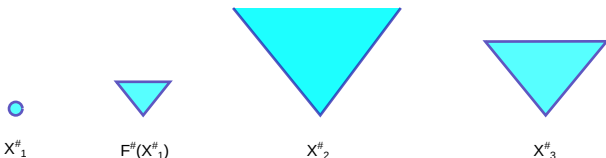
Example analysis (illustration)

Example program

```

X ← 2; I ← 0;
while • I < 10 do
  if [0,1] = 0 then X ← X + 2 else X ← X - 3 fi;
  I ← I + 1
done ♦

```



$$X_1^\# = \{X = 2, I = 0\}$$

$$\begin{aligned}
 X_2^\# &= \{X = 2, I = 0\} \nabla (\{X = 2, I = 0\} \cup^\# \{X \in [-1, 4], I = 1\}) \\
 &= \{I \geq 0, 2 - 3I \leq X \leq 2I + 2\}
 \end{aligned}$$

$$\begin{aligned}
 X_3^\# &= \{X = 2, I = 0\} \cup^\# \{I \in [1, 10], 2 - 3I \leq X \leq 2I + 2\} \\
 &= \{I \in [0, 10], 2 - 3I \leq X \leq 2I + 2\}
 \end{aligned}$$

Other polyhedra widenings

Widening with thresholds:

Given a **finite** set T of **constraints**, we add to $\mathcal{X}^\# \nabla \mathcal{Y}^\#$ all the constraints from T satisfied by both $\mathcal{X}^\#$ and $\mathcal{Y}^\#$.

Delayed widening:

We replace $\mathcal{X}^\# \nabla \mathcal{Y}^\#$ with $\mathcal{X}^\# \cup^\# \mathcal{Y}^\#$ a **finite** number of times (this works for any widening and abstract domain).

See also [Bagn03].

Strict inequalities

The polyhedron domain can be extended to allow strict constraints: $\{ \vec{V} \mid \mathbf{M} \times \vec{V} \geq \vec{C} \text{ and } \mathbf{M}' \times \vec{V} > \vec{C}' \}$

Idea:

A **non-closed** polyhedron on \mathbb{V} is represented as a **closed** polyhedron P on $\mathbb{V}' \stackrel{\text{def}}{=} \mathbb{V} \cup \{V_\epsilon\}$.

$$\begin{array}{ll} \alpha_1 V_1 + \dots + \alpha_n V_n + 0 V_\epsilon \geq 0 & \text{represents } \alpha_1 V_1 + \dots + \alpha_n V_n \geq 0 \\ \alpha_1 V_1 + \dots + \alpha_n V_n - c V_\epsilon \geq 0, c > 0 & \text{represents } \alpha_1 V_1 + \dots + \alpha_n V_n > 0 \end{array}$$

P represents the non necessarily closed polyhedron:

$$\gamma_\epsilon(P) \stackrel{\text{def}}{=} \{(V_1, \dots, V_n) \mid \exists V_\epsilon > 0, (V_1, \dots, V_n, V_\epsilon) \in \gamma(P)\}.$$

Notes:

- The minimal form needs some adaptation [Bagn02].
- Chernikova's algorithm, \cap^\sharp , \cup^\sharp , $C^\sharp[[c]]$, and $\overleftarrow{C}^\sharp[[c]]$ can be easily reused.

Integer polyhedra

How can we deal with $\mathbb{I} = \mathbb{Z}$?

Issue: integer linear programming is difficult.

Example: satisfiability of conjunctions of linear constraints:

- polynomial cost in \mathbb{Q} ,
- NP-complete cost in \mathbb{Z} .

Possible solutions:

- Use some complete integer algorithms.
(e.g. Presburger arithmetics)
Costly, and we do not have any abstract domain structure.
- Keep \mathbb{Q} -polyhedra as representation, and change the concretization into: $\gamma_{\mathbb{Z}}(\mathcal{X}^{\#}) \stackrel{\text{def}}{=} \gamma(\mathcal{X}^{\#}) \cap \mathbb{Z}^n$.
However, operators are no longer exact / optimal.

Weakly relational domains

Zone domain

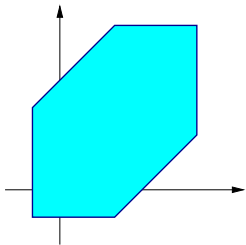
The zone domain

Here, $\mathbb{I} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$.

We look for invariants of the form:

$$\bigwedge V_i - V_j \leq c \text{ or } \pm V_i \leq c, \quad c \in \mathbb{I}$$

A subset of \mathbb{I}^n bounded by such constraints is called a **zone**.



[Mine01a]

Machine representation

A **potential constraint** has the form: $V_j - V_i \leq c$.

Potential graph: directed, weighted graph \mathcal{G}

- nodes are labelled with variables in \mathbb{V} ,
- we add an arc with **weight** c from V_i to V_j for each constraint $V_j - V_i \leq c$.

Difference Bound Matrix (DBM)

Adjacency matrix \mathbf{m} of \mathcal{G} :

- \mathbf{m} is square, with size $n \times n$, and elements in $\mathbb{I} \cup \{+\infty\}$,
- $m_{ij} = c < +\infty$ denotes the constraint $V_j - V_i \leq c$,
- $m_{ij} = +\infty$ if there is no upper bound on $V_j - V_i$.

Concretization:

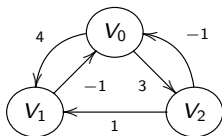
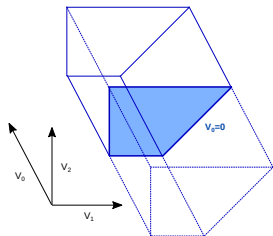
$$\gamma(\mathbf{m}) \stackrel{\text{def}}{=} \{ (v_1, \dots, v_n) \in \mathbb{I}^n \mid \forall i, j, v_j - v_i \leq m_{ij} \}.$$

Machine representation (cont.)

Unary constraints add a constant null variable V_0 .

- \mathbf{m} has size $(n + 1) \times (n + 1)$;
- $V_i \leq c$ is denoted as $V_i - V_0 \leq c$, i.e., $m_{i0} = c$;
- $V_i \geq c$ is denoted as $V_0 - V_i \leq -c$, i.e., $m_{0i} = -c$;
- γ is now: $\gamma_0(\mathbf{m}) \stackrel{\text{def}}{=} \{ (v_1, \dots, v_n) \mid (0, v_1, \dots, v_n) \in \gamma(\mathbf{m}) \}$.

Example:



| | V_0 | V_1 | V_2 |
|-------|-----------|-----------|-----------|
| V_0 | $+\infty$ | 4 | 3 |
| V_1 | -1 | $+\infty$ | $+\infty$ |
| V_2 | -1 | 1 | $+\infty$ |

The DBM lattice

$\mathcal{D}^\#$ contains all DBMs, plus $\perp^\#$.

\leq on $\mathbb{I} \cup \{+\infty\}$ is extended **point-wisely**.

If $\mathbf{m}, \mathbf{n} \neq \perp^\#$:

$$\begin{array}{lll}
 \mathbf{m} \subseteq^\# \mathbf{n} & \stackrel{\text{def}}{\iff} & \forall i, j, m_{ij} \leq n_{ij} \\
 \mathbf{m} =^\# \mathbf{n} & \stackrel{\text{def}}{\iff} & \forall i, j, m_{ij} = n_{ij} \\
 [\mathbf{m} \cap^\# \mathbf{n}]_{ij} & \stackrel{\text{def}}{=} & \min(m_{ij}, n_{ij}) \\
 [\mathbf{m} \cup^\# \mathbf{n}]_{ij} & \stackrel{\text{def}}{=} & \max(m_{ij}, n_{ij}) \\
 [\top^\#]_{ij} & \stackrel{\text{def}}{=} & +\infty
 \end{array}$$

$(\mathcal{D}^\#, \subseteq^\#, \cup^\#, \cap^\#, \perp^\#, \top^\#)$ is a **lattice**.

Remarks:

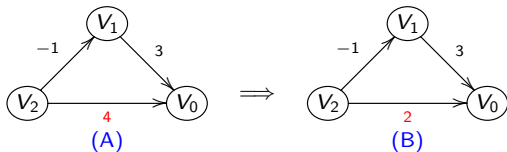
- $\mathcal{D}^\#$ is complete if \leq is ($\mathbb{I} = \mathbb{R}$ or \mathbb{Z} , but not \mathbb{Q}),
- $\mathbf{m} \subseteq^\# \mathbf{n} \implies \gamma_0(\mathbf{m}) \subseteq \gamma_0(\mathbf{n})$, but **not the converse**,
- $\mathbf{m} =^\# \mathbf{n} \implies \gamma_0(\mathbf{m}) = \gamma_0(\mathbf{n})$, but **not the converse**.

Normal form, equality and inclusion testing

Issue: how can we compare $\gamma_0(\mathbf{m})$ and $\gamma_0(\mathbf{n})$?

Idea: find a normal form by **propagating/tightening constraints**.

$$\left\{ \begin{array}{l} V_0 - V_1 \leq 3 \\ V_1 - V_2 \leq -1 \\ V_0 - V_2 \leq 4 \end{array} \right. \quad \Rightarrow \quad \left\{ \begin{array}{l} V_0 - V_1 \leq 3 \\ V_1 - V_2 \leq -1 \\ V_0 - V_2 \leq 2 \end{array} \right.$$



Definition: shortest-path closure \mathbf{m}^*

$$m_{ij}^* \stackrel{\text{def}}{=} \min_N \sum_{k=1}^{N-1} m_{i_k i_{k+1}} \langle i = i_1, \dots, i_N = j \rangle$$

Exists only when \mathbf{m} has no cycle with strictly negative weight.

Floyd–Warshall algorithm

Properties:

- $\gamma_0(\mathbf{m}) = \emptyset \iff \mathcal{G}$ has a cycle with strictly negative weight.
- if $\gamma_0(\mathbf{m}) \neq \emptyset$, the shortest-path graph \mathbf{m}^* is a normal form:

$$\mathbf{m}^* = \min_{\subseteq^\#} \{ \mathbf{n} \mid \gamma_0(\mathbf{m}) = \gamma_0(\mathbf{n}) \}$$
- If $\gamma_0(\mathbf{m}), \gamma_0(\mathbf{n}) \neq \emptyset$, then
 - $\gamma_0(\mathbf{m}) = \gamma_0(\mathbf{n}) \iff \mathbf{m}^* =^\# \mathbf{n}^*$,
 - $\gamma_0(\mathbf{m}) \subseteq \gamma_0(\mathbf{n}) \iff \mathbf{m}^* \subseteq^\# \mathbf{n}$.

Floyd–Warshall algorithm

$$\begin{cases} m_{ij}^0 & \stackrel{\text{def}}{=} & m_{ij} \\ m_{ij}^{k+1} & \stackrel{\text{def}}{=} & \min(m_{ij}^k, m_{ik}^k + m_{kj}^k) \end{cases}$$

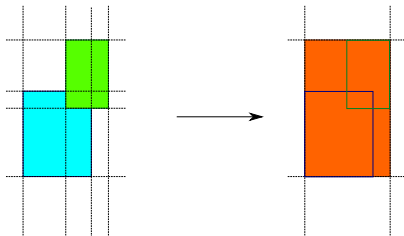
- If $\gamma_0(\mathbf{m}) \neq \emptyset$, then $\mathbf{m}^* = \mathbf{m}^{n+1}$, (normal form)
- $\gamma_0(\mathbf{m}) = \emptyset \iff \exists i, m_{ii}^{n+1} < 0$, (emptiness testing)
- \mathbf{m}^{n+1} can be computed in $\mathcal{O}(n^3)$ time.

Abstract operators

Abstract join: naïve version \cup^\sharp (*element-wise max*)

- \cup^\sharp is a **sound abstraction** of \cup

but $\gamma_0(\mathbf{m} \cup^\sharp \mathbf{n})$ is **not necessarily the smallest zone** containing $\gamma_0(\mathbf{m})$ and $\gamma_0(\mathbf{n})$!



The union of two zones with \cup^\sharp is no more precise in the zone domain than in the interval domain!

Abstract operators (cont.)

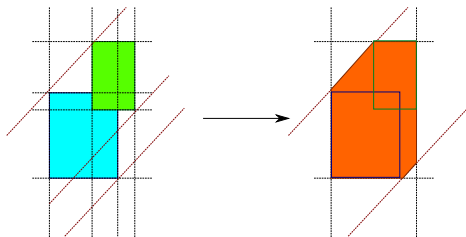
Abstract join: precise version: \cup^\sharp after closure

- $(\mathbf{m}^*) \cup^\sharp (\mathbf{n}^*)$ is however **optimal**

we have: $(\mathbf{m}^*) \cup^\sharp (\mathbf{n}^*) = \min_{\subseteq^\sharp} \{ \mathbf{o} \mid \gamma_0(\mathbf{o}) \supseteq \gamma_0(\mathbf{m}) \cup \gamma_0(\mathbf{n}) \}$

which implies:

$$\gamma_0((\mathbf{m}^*) \cup^\sharp (\mathbf{n}^*)) = \min_{\subseteq} \{ \gamma_0(\mathbf{o}) \mid \gamma_0(\mathbf{o}) \supseteq \gamma_0(\mathbf{m}) \cup \gamma_0(\mathbf{n}) \}$$



after closure, new constraints $c \leq X - Y \leq d$ give an increase in precision

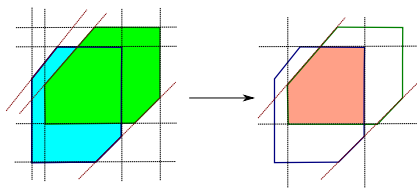
- $(\mathbf{m}^*) \cup^\sharp (\mathbf{n}^*)$ is always closed.

Abstract operators (cont.)

Abstract intersection $\cap^\#$: element-wise min

- $\cap^\#$ is an exact abstraction of \cap (zones are closed under intersection):

$$\gamma_0(\mathbf{m} \cap^\# \mathbf{n}) = \gamma_0(\mathbf{m}) \cap \gamma_0(\mathbf{n})$$



- $(\mathbf{m}^*) \cap^\# (\mathbf{n}^*)$ is not necessarily closed...

Remark

The set of closed matrices, with $\perp^\#$, and the operations $\subseteq^\#, \cup^\#, \lambda \mathbf{m}, \mathbf{n}. (\mathbf{m} \cap^\# \mathbf{n})^*$ is a sub-lattice, where γ_0 is injective.

Abstract operators (cont.)

We can define:

$$[C^\sharp \llbracket V_{j_0} - V_{i_0} \leq c \rrbracket \mathbf{m}]_{ij} \stackrel{\text{def}}{=} \begin{cases} \min(m_{ij}, c) & \text{if } (i, j) = (i_0, j_0), \\ m_{ij} & \text{otherwise.} \end{cases}$$

$$C^\sharp \llbracket V_{j_0} - V_{i_0} = [a, b] \rrbracket \mathbf{m} \stackrel{\text{def}}{=} (C^\sharp \llbracket V_{j_0} - V_{i_0} \leq b \rrbracket \circ C^\sharp \llbracket V_{i_0} - V_{j_0} \leq -a \rrbracket) \mathbf{m}$$

$$[C^\sharp \llbracket V_{j_0} \leftarrow [-\infty, +\infty] \rrbracket \mathbf{m}]_{ij} \stackrel{\text{def}}{=} \begin{cases} +\infty & \text{if } i = j_0 \text{ or } j = j_0, \\ m_{ij}^* & \text{otherwise.} \end{cases}$$

(not optimal on non-closed arguments)

$$C^\sharp \llbracket V_{j_0} \leftarrow V_{i_0} + [a, b] \rrbracket \mathbf{m} \stackrel{\text{def}}{=} (C^\sharp \llbracket V_{j_0} - V_{i_0} = [a, b] \rrbracket \circ C^\sharp \llbracket V_{j_0} \leftarrow [-\infty, +\infty] \rrbracket) \mathbf{m} \quad \text{if } i_0 \neq j_0$$

$$[C^\sharp \llbracket V_{j_0} \leftarrow V_{j_0} + [a, b] \rrbracket \mathbf{m}]_{ij} \stackrel{\text{def}}{=} \begin{cases} m_{ij} - a & \text{if } i = j_0 \text{ and } j \neq j_0 \\ m_{ij} + b & \text{if } i \neq j_0 \text{ and } j = j_0 \\ m_{ij} & \text{otherwise.} \end{cases}$$

($i_0 \neq j_0$; V_{i_0} can be replaced with 0 by setting $i_0 = 0$)

These transfer functions are **exact**.

Abstract operators (cont.)

Backward assignment:

$$\overleftarrow{C}^\# \llbracket V_{j_0} \leftarrow [-\infty, +\infty] \rrbracket (\mathbf{m}, \mathbf{r}) \stackrel{\text{def}}{=} \mathbf{m} \cap^\# (C^\# \llbracket V_{j_0} \leftarrow [-\infty, +\infty] \rrbracket \mathbf{r})$$

$$\overleftarrow{C}^\# \llbracket V_{j_0} \leftarrow V_{j_0} + [a, b] \rrbracket (\mathbf{m}, \mathbf{r}) \stackrel{\text{def}}{=} \mathbf{m} \cap^\# (C^\# \llbracket V_{j_0} \leftarrow V_{j_0} + [-b, -a] \rrbracket \mathbf{r})$$

$$\left[\overleftarrow{C}^\# \llbracket V_{j_0} \leftarrow V_{i_0} + [a, b] \rrbracket (\mathbf{m}, \mathbf{r}) \right]_{ij} \stackrel{\text{def}}{=} \mathbf{m} \cap^\# \begin{cases} \min(\mathbf{r}_{ij}^*, \mathbf{r}_{j_0j}^* + b) & \text{if } i = i_0 \text{ and } j \neq i_0, j_0 \\ \min(\mathbf{r}_{ij}^*, \mathbf{r}_{ij_0}^* - a) & \text{if } j = i_0 \text{ and } i \neq i_0, j_0 \\ +\infty & \text{if } i = j_0 \text{ or } j = j_0 \\ \mathbf{r}_{ij}^* & \text{otherwise.} \end{cases}$$

Abstract operators (cont.)

Issue: given an arbitrary linear assignment $V_{j_0} \leftarrow a_0 + \sum_k a_k \times V_k$

- there is no exact abstraction, in general;
- the best abstraction $\alpha \circ C[[c]] \circ \gamma$ is costly to compute.
(e.g. convert to a polyhedron and back, with exponential cost)

Possible solution:

Given a (more general) assignment $e = [a_0, b_0] + \sum_k [a_k, b_k] \times V_k$

we define an **approximate** operator as follows:

$$[C^\#[[V_{j_0} \leftarrow e]]\mathbf{m}]_{ij} \stackrel{\text{def}}{=} \begin{cases} \max(E^\#[[e]]\mathbf{m}) & \text{if } i = 0 \text{ and } j = j_0 \\ -\min(E^\#[[e]]\mathbf{m}) & \text{if } i = j_0 \text{ and } j = 0 \\ \max(E^\#[[e - V_i]]\mathbf{m}) & \text{if } i \neq 0, j_0 \text{ and } j = j_0 \\ -\min(E^\#[[e + V_j]]\mathbf{m}) & \text{if } i = j_0 \text{ and } j \neq 0, j_0 \\ m_{ij} & \text{otherwise} \end{cases}$$

where $E^\#[[e]]\mathbf{m}$ evaluates e using interval arithmetics with $V_k \in [-m_{k0}^*, m_{0k}^*]$.

Quadratic total cost (plus the cost of closure).

Abstract operators (cont.)

Example:

Argument

$$\left\{ \begin{array}{l} 0 \leq Y \leq 10 \\ 0 \leq Z \leq 10 \\ 0 \leq Y - Z \leq 10 \end{array} \right.$$

$$\Downarrow X \leftarrow Y - Z$$

$$\left\{ \begin{array}{l} -10 \leq X \leq 10 \\ -20 \leq X - Y \leq 10 \\ -20 \leq X - Z \leq 10 \end{array} \right.$$

Intervals

$$\left\{ \begin{array}{l} -10 \leq X \leq 10 \\ -10 \leq X - Y \leq 0 \\ -10 \leq X - Z \leq 10 \end{array} \right.$$

Approximate
solution

$$\left\{ \begin{array}{l} 0 \leq X \leq 10 \\ -10 \leq X - Y \leq 0 \\ -10 \leq X - Z \leq 10 \end{array} \right.$$

Best
(polyhedra)

We have a good trade-off between cost and precision.

The same idea can be used for tests and backward assignments.

Widening and narrowing

The zone domain has both strictly increasing and decreasing infinite chains.

Widening ∇

$$[\mathbf{m} \nabla \mathbf{n}]_{ij} \stackrel{\text{def}}{=} \begin{cases} m_{ij} & \text{if } n_{ij} \leq m_{ij} \\ +\infty & \text{otherwise} \end{cases}$$

Unstable constraints are deleted.

Narrowing Δ

$$[\mathbf{m} \Delta \mathbf{n}]_{ij} \stackrel{\text{def}}{=} \begin{cases} n_{ij} & \text{if } m_{ij} = +\infty \\ m_{ij} & \text{otherwise} \end{cases}$$

Only $+\infty$ bounds are refined.

Remarks:

- We can construct widenings with thresholds.
- ∇ (resp. Δ) can be seen as a **point-wise extension** of an interval widening (resp. narrowing).

Interaction between closure and widening

Widening ∇ and closure $*$ cannot always be mixed safely:

- $\mathbf{m}_{i+1} \stackrel{\text{def}}{=} \mathbf{m}_i \nabla (\mathbf{n}_i^*)$ OK
- $\mathbf{m}_{i+1} \stackrel{\text{def}}{=} (\mathbf{m}_i^*) \nabla \mathbf{n}_i$ wrong!
- $\mathbf{m}_{i+1} \stackrel{\text{def}}{=} (\mathbf{m}_i \nabla \mathbf{n}_i)^*$ wrong

otherwise the sequence (\mathbf{m}_i) may be infinite!

Example:

```

X ← 0; Y ← [-1,1];
while • 1 = 1 do
  R ← [-1,1];
  if X = Y then Y ← X + R
  else X ← Y + R fi
done
  
```

| iter. | X | Y | X - Y |
|----------|---------------------|---------------------|-----------|
| 0 | 0 | [-1, 1] | [-1, 1] |
| 1 | [-2, 2] | [-1, 1] | [-1, 1] |
| 2 | [-2, 2] | [-3, 3] | [-1, 1] |
| ... | ... | ... | ... |
| $2j$ | $[-2j, 2j]$ | $[-2j - 1, 2j + 1]$ | $[-1, 1]$ |
| $2j + 1$ | $[-2j - 2, 2j + 2]$ | $[-2j - 1, 2j + 1]$ | $[-1, 1]$ |

Applying the closure after the widening at • prevents convergence.
Without the closure, we would find in finite time $X - Y \in [-1, 1]$.

Note: this situation also occurs in **reduced products**.

(here, $\mathcal{D}^\# \simeq$ reduced product of $n \times n$ intervals, $*$ \simeq reduction)

Interaction between closure and widening (illustration)

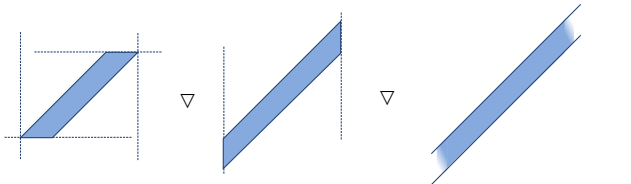
```

X ← 0; Y ← [-1,1];
while 1 = 1 do
  R ← [-1,1];
  if X = Y then Y ← X + R
  else X ← Y + R fi
done

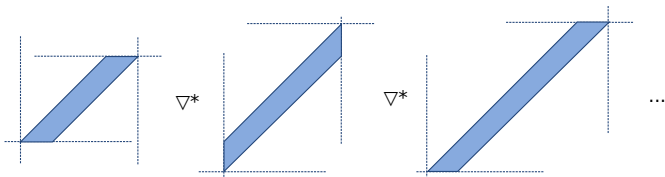
```

| iter. | X | Y | X - Y |
|--------|---------------------|---------------------|---------|
| 0 | 0 | [-1, 1] | [-1, 1] |
| 1 | [-2, 2] | [-1, 1] | [-1, 1] |
| 2 | [-2, 2] | [-3, 3] | [-1, 1] |
| ... | ... | ... | ... |
| 2j | $[-2j, 2j]$ | $[-2j - 1, 2j + 1]$ | [-1, 1] |
| 2j + 1 | $[-2j - 2, 2j + 2]$ | $[-2j - 1, 2j + 1]$ | [-1, 1] |

widening
without
closure



widening
with
closure



Octagon domain

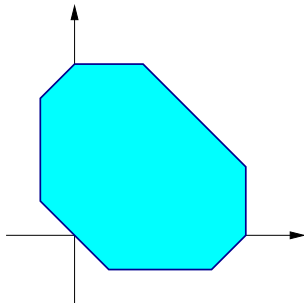
The octagon domain

Now, $\mathbb{I} \in \{\mathbb{Q}, \mathbb{R}\}$.

We look for invariants of the form: $\bigwedge \pm V_i \pm V_j \leq c, \quad c \in \mathbb{I}$

A subset of \mathbb{I}^n defined by such constraints is called an **octagon**.

It is a generalisation of zones (more symmetric).



[Mine01b]

Machine representation

Idea: use a **variable change** to get back to potential constraints.

Let $\mathbb{V}' \stackrel{\text{def}}{=} \{V'_1, \dots, V'_{2n}\}$.

| the constraint: | is encoded as: |
|--------------------------------------|--|
| $V_i - V_j \leq c \quad (i \neq j)$ | $V'_{2i-1} - V'_{2j-1} \leq c \quad \text{and} \quad V'_{2j} - V'_{2i} \leq c$ |
| $V_i + V_j \leq c \quad (i \neq j)$ | $V'_{2i-1} - V'_{2j} \leq c \quad \text{and} \quad V'_{2j-1} - V'_{2i} \leq c$ |
| $-V_i - V_j \leq c \quad (i \neq j)$ | $V'_{2j} - V'_{2i-1} \leq c \quad \text{and} \quad V'_{2i} - V'_{2j-1} \leq c$ |
| $V_i \leq c$ | $V'_{2i-1} - V'_{2i} \leq 2c$ |
| $V_i \geq c$ | $V'_{2i} - V'_{2i-1} \leq -2c$ |

We use a matrix \mathbf{m} of size $(2n) \times (2n)$ with elements in $\mathbb{I} \cup \{+\infty\}$ and $\gamma_{\pm}(\mathbf{m}) \stackrel{\text{def}}{=} \{(v_1, \dots, v_n) \mid (v_1, -v_1, \dots, v_n, -v_n) \in \gamma(\mathbf{m})\}$.

Note:

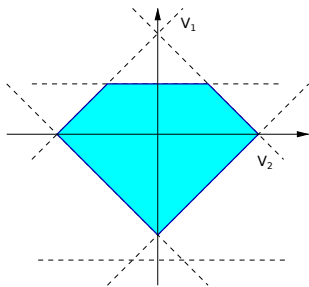
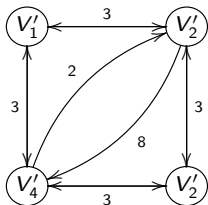
Two distinct \mathbf{m} elements can represent the same constraint on \mathbb{V} .

To avoid this, we impose that $\forall i, j, m_{ij} = m_{j\bar{i}}$ where $\bar{i} = i \oplus 1$.

Machine representation (cont.)

Example:

$$\left\{ \begin{array}{l} V_1 + V_2 \leq 3 \\ V_2 - V_1 \leq 3 \\ V_1 - V_2 \leq 3 \\ -V_1 - V_2 \leq -3 \\ 2V_2 \leq 2 \\ -2V_2 \leq 8 \end{array} \right.$$



Lattice

Constructed by point-wise extension of \leq on $\mathbb{N} \cup \{+\infty\}$.

Algorithms

m^* is not a normal form for γ_{\pm} .

Idea use **two** local transformations instead of one:

$$\begin{cases} V'_i - V'_k \leq c \\ V'_k - V'_j \leq d \end{cases} \implies V'_i - V'_j \leq c + d$$

and

$$\begin{cases} V'_i - V'_{\bar{i}} \leq c \\ V'_{\bar{j}} - V'_j \leq d \end{cases} \implies V'_i - V'_j \leq (c + d)/2$$

Modified Floyd–Warshall algorithm

$$m^{\bullet} \stackrel{\text{def}}{=} S(m^{2n+1})$$

where:

$$(A) \quad \begin{cases} m^1 \stackrel{\text{def}}{=} m \\ [m^{k+1}]_{ij} \stackrel{\text{def}}{=} \min(n_{ij}, n_{ik} + n_{kj}), 1 \leq k \leq 2n \end{cases}$$

$$(B) \quad [S(n)]_{ij} \stackrel{\text{def}}{=} \min(n_{ij}, (n_{i\bar{i}} + n_{\bar{j}j})/2)$$

Algorithms (cont.)

Applications

- $\gamma_{\pm}(\mathbf{m}) = \emptyset \iff \exists i, \mathbf{m}_{ii}^{\bullet} < 0,$
- if $\gamma_{\pm}(\mathbf{m}) \neq \emptyset$, \mathbf{m}^{\bullet} is a normal form:

$$\mathbf{m}^{\bullet} = \min_{\subseteq^{\#}} \{ \mathbf{n} \mid \gamma_{\pm}(\mathbf{n}) = \gamma_{\pm}(\mathbf{m}) \},$$
- $(\mathbf{m}^{\bullet}) \cup^{\#} (\mathbf{n}^{\bullet})$ is the best abstraction for the set-union
 $\gamma_{\pm}(\mathbf{m}) \cup \gamma_{\pm}(\mathbf{n}).$

Widening and narrowing

- The zone widening and narrowing can be used on octagons.
- The widened iterates should not be closed.
 (prevents convergence)

Abstract transfer functions are similar to the case of the zone domain.

Analysis example

Rate limiter

```

Y ← 0; while 1=1 do
  X ← [-128,128]; D ← [0,16];
  S ← Y; Y ← X; R ← X - S;
  if R ≤ -D then Y ← S - D fi;
  if R ≥ D then Y ← S + D fi
done

```

```

X:  input signal
Y:  output signal
S:  last output
R:  delta Y - S
D:  max. allowed for |R|

```

Analysis using:

- the octagon domain,
- an abstract operator for $V_{j_0} \leftarrow [a_0, b_0] + \sum_k [a_k, b_k] \times V_k$ similar to the one we defined on zones,
- a widening with thresholds T .

Result: we prove that $|Y|$ is bounded by: $\min \{ t \in T \mid t \geq 144 \}$.

Note: the polyhedron domain would find $|Y| \leq 128$ and does not require thresholds, but it is more costly.

Summary

Summary of numerical domains

| domain | invariants | memory cost | time cost (per operation) |
|-------------------|------------------------------------|------------------------------------|---------------------------|
| intervals | $V \in [\ell, h]$ | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ |
| linear equalities | $\sum_i \alpha_i V_i = \beta_i$ | $\mathcal{O}(n ^2)$ | $\mathcal{O}(n ^3)$ |
| zones | $V_i - V_j \leq c$ | $\mathcal{O}(n ^2)$ | $\mathcal{O}(n ^3)$ |
| polyhedra | $\sum_i \alpha_i V_i \geq \beta_i$ | unbounded, exponential in practice | |

- abstract domains provide trade-offs between cost and precision
- **relational invariants** are often necessary
even to prove non-relational properties
- an abstract domain is defined by the choice of:
 - some **properties of interest** and **operators** *(semantic part)*
 - **data-structures** and **algorithms** *(algorithmic part)*
- an analysis mixes two kinds of approximations:
 - **static** approximations *(choice of abstract properties)*
 - **dynamic** approximations *(widening)*

Bibliography

Bibliography

[Anco10] **C. Ancourt, F. Coelho & F. Irigoien.** *A modular static analysis approach to affine loop invariants detection.* In Proc. NSAD'10, ENTCS, Elsevier, 2010.

[Bagn02] **R. Bagnara, E. Ricci, E. Zaffanella & P. M. Hill.** *Possibly not closed convex polyhedra and the Parma Polyhedra Library.* In Proc. SAS'02, LNCS 2477, 213–229, Springer, 2002.

[Bagn03] **R. Bagnara, P. Hill, E. Ricci, E. Zaffanella.** *Precise widening operators for convex polyhedra.* In Proc. SAS'03, LNCS 2694, 337-354, Springer, 2003.

[Bagn08] **R. Bagnara, P. M. Hill & E. Zaffanella.** *An improved tight closure algorithm for integer octagonal constraints.* In Proc. VMCAI'08, LNCS 4905, 8–21, Springer, 2008.

[Beno96] **F. Benoy & A. King.** *Inferring argument size relationships with CLP(R).* In In Proc. of LOPSTR'96, LNCS 1207, 204–223. Springer, 1996.

Bibliography (cont.)

[Cher68] **N. V. Chernikova**. *Algorithm for discovering the set of all the solutions of a linear programming problem*. In U.S.S.R. Comput. Math. and Math. Phys., 8(6):282–293, 1968.

[Cous78] **P. Cousot & N. Halbwachs**. *Automatic discovery of linear restraints among variables of a program*. In Proc. POPL'78, 84–96, ACM, 1978.

[Gran91] **P. Granger**. *Static analysis of linear congruence equalities among variables of a program*. In Proc. TAPSOFT'91, LNCS 49, 169–192. Springer, 1991.

[Jean09] **B. Jeannet & A. Miné**. *Apron: A library of numerical abstract domains for static analysis*. In Proc. CAV'09, LNCS 5643, 661–667, Springer, 2009, <http://apron.cri.enscm.fr/library>.

Bibliography (cont.)

[Karr76] **M. Karr**. *Affine relationships among variables of a program*. In Acta Informatica, 6:133–151, 1976.

[LeVe92] **H. Le Verge**. *A note on Chernikova's algorithm*. In Research Report 1662, INRIA Rocquencourt, 1992.

[Mine01a] **A. Miné**. *A new numerical abstract domain based on difference-bound matrices*. In Proc. PADO II, LNCS 2053, 155–172, Springer, 2001.

[Mine01b] **A. Miné**. *The octagon abstract domain*. In Proc. AST'01, 310–319, IEEE, 2001.

[Schr86] **A. Schrijver**. *Theory of linear and integer programming*. In John Wiley & Sons, Inc., 1986.