New Algorithmics for Polyhedral Calculus via Parametric Linear Programming

Alexandre Maréchal

Thèse réalisée à VERIMAG sous la direction de
David Monniaux et Michaël Périn

GDR GPL 2018 - Accessit au prix de thèse
13 Juin 2018

```
// x ∈ [0,5]
// y ∈ [-5,5]
int z;
if (2*x + 8*y >= 11){
    z = 1/y;
}
```

```
// x ∈ [0,5]
// y ∈ [−5,5]
int z;
if (2*x + 8*y >= 11){
    z = 1/y;
}
```

Can y be 0?

```
// x ∈ [0,5]
// y ∈ [−5,5]
int z;
if (2*x + 8*y >= 11){
    z = 1/y;
}
```

Can y be 0?

$$\begin{aligned} & 0 \le x \le 5 \\ \wedge \quad & -5 \le y \le 5 \\ \wedge \quad & 2x + 8y \ge 11 \\ \wedge \quad & y = 0 \end{aligned}$$

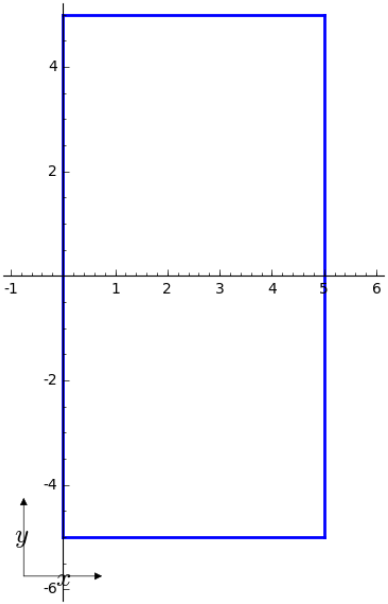
Is this feasible ?

```
// x ∈ [0,5]
// y ∈ [−5,5]
int z;
if (2*x + 8*y >= 11){
    z = 1/y;
}
```

Can y be 0?

$$\begin{array}{ll} & 0 \le x \le 5 \\ \wedge & -5 \le y \le 5 \\ \wedge & 2x + 8y \ge 11 \\ \wedge & y = 0 \end{array}$$

Is this feasible ?

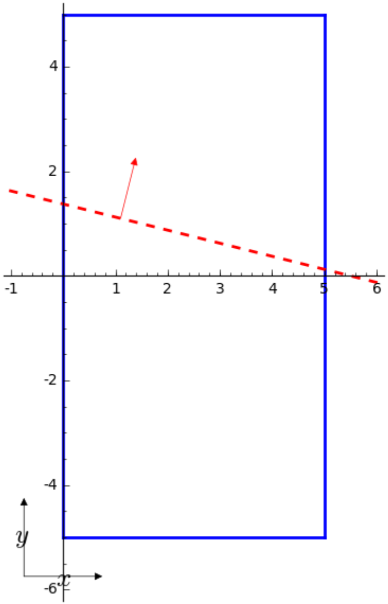


```
// x ∈ [0,5]
// y ∈ [−5,5]
int z;
if (2*x + 8*y >= 11){
      z = 1/y;
}
```

Can y be 0?

$$\begin{array}{ll} & 0 \leq x \leq 5 \\ \wedge & -5 \leq y \leq 5 \\ \wedge & 2x + 8y \geq 11 \\ \wedge & y = 0 \end{array}$$

Is this feasible ?

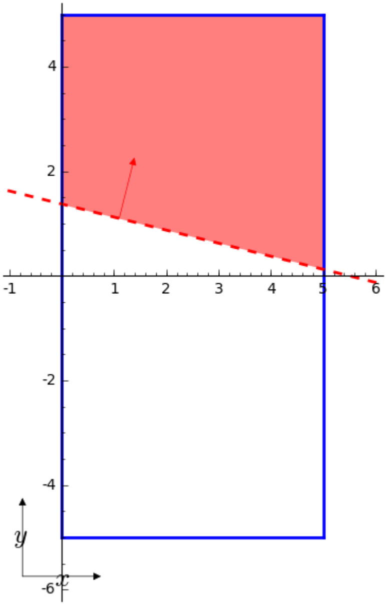


```
// x ∈ [0,5]
// y ∈ [−5,5]
int z;
if (2*x + 8*y >= 11){
    z = 1/y;
}
```

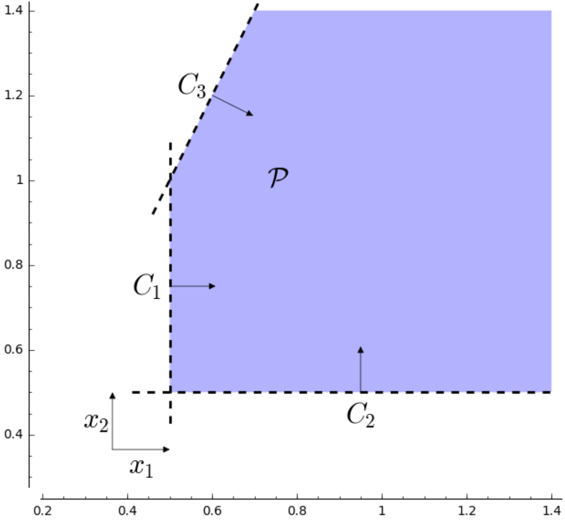
Can y be 0?

$$\begin{array}{rl} & 0 \leq x \leq 5 \\ \wedge & -5 \leq y \leq 5 \\ \wedge & 2x + 8y \geq 11 \\ \wedge & y = 0 \end{array}$$

Is this feasible ?



As **constraints**

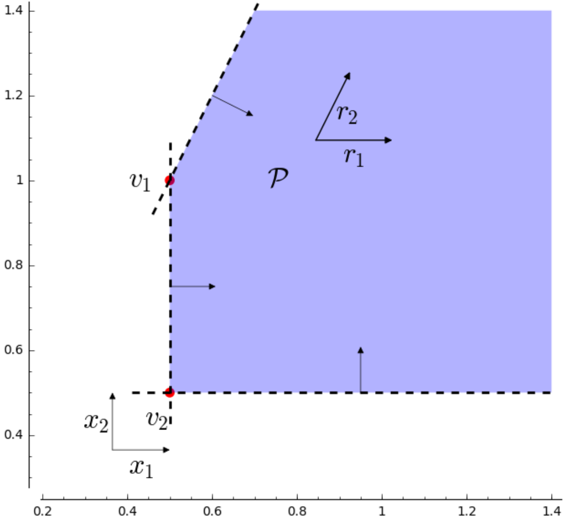


$\boldsymbol{C}_1 : 2x_1 \geq 1$

$\boldsymbol{C}_2 : 2x_2 \geq 1$

$\boldsymbol{C}_3 : 2x_1 - x_2 \geq 0$

As **generators**: vertices and rays



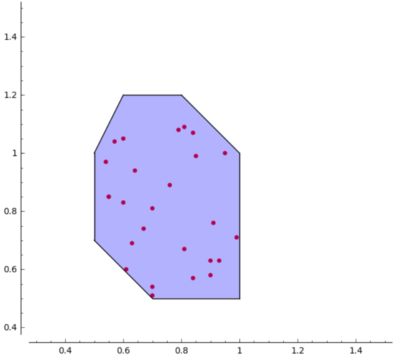
$\boldsymbol{v_1} : \left(\frac{1}{2}, 1\right)$

$\boldsymbol{v_2} : \left(\frac{1}{2}, \frac{1}{2}\right)$

$\boldsymbol{r}_1 : (1, 0)$

$\boldsymbol{r}_2 : (1, 1)$

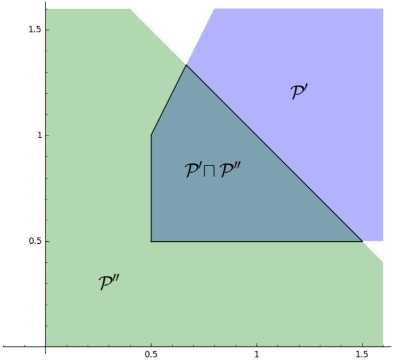
Abstract Interpretation: prove program properties by over-approximating reachable memory states



The Abstract domain of polyhedra [Cousot and Halbwachs, 1978]

- handles affine relations between variables;
- is powerful, but expensive

$$
\begin{array}{l}
\dots \\
/*\ \ (x_1, x_2) \in \mathcal{P}'\ \ */ \\
\texttt{if}\ \ (x_1 + x_2 \le 2)\{ \\
\qquad\qquad /*\ \ (x_1, x_2) \in \mathcal{P}' \sqcap \{x_1 + x_2 \le 2\}\ \ */ \\
\}
\end{array}
$$



$$
\mathcal{P}' : \begin{cases} \boldsymbol{C}_1 : 2x_1 \ge 1 \\ \boldsymbol{C}_2 : 2x_2 \ge 1 \\ \boldsymbol{C}_3 : 2x_1 - x_2 \ge 0 \end{cases}
$$

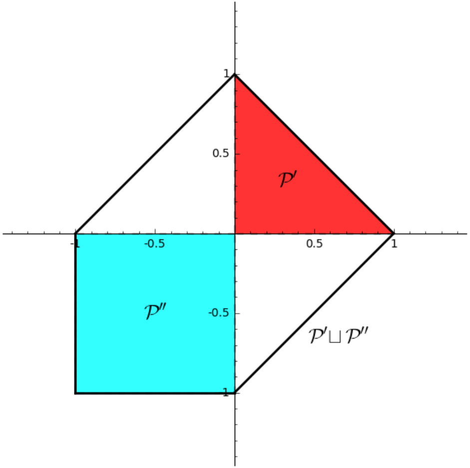
$$
\mathcal{P}'' : \left\{ \boldsymbol{C}' : x_1 + x_2 \le 2 \right.
$$

```
...
/*
if (...){
        ...
        (x1, x2) ∈ P'  */
}
else{
        ...
        (x1, x2) ∈ P''  */
}
/* (x1, x2) ∈ P' ⊔ P''  */
```

$(x_1, x_2) \in \mathcal{P}'$

$(x_1, x_2) \in \mathcal{P}''$

$(x_1, x_2) \in \mathcal{P}' \sqcup \mathcal{P}''$



Static Analyzers aim at verifying programs. For critical software

- false alarms are acceptable
- missed errors are not

Static Analyzers aim at verifying programs. For critical software

- false alarms are acceptable
- missed errors are not

But analyzers can be buggy (maybe more than other programs!):

- Written by students
- Not tested as much as critical programs

Static Analyzers aim at verifying programs. For critical software

- false alarms are acceptable
- missed errors are not

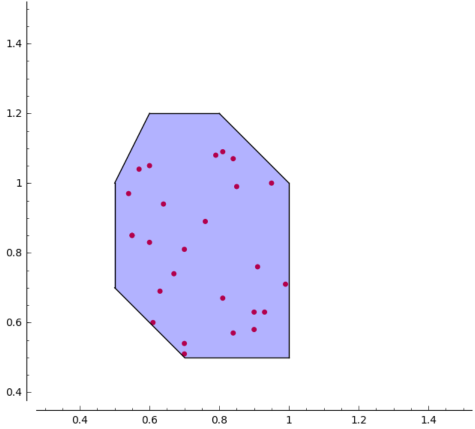
But analyzers can be buggy (maybe more than other programs!):

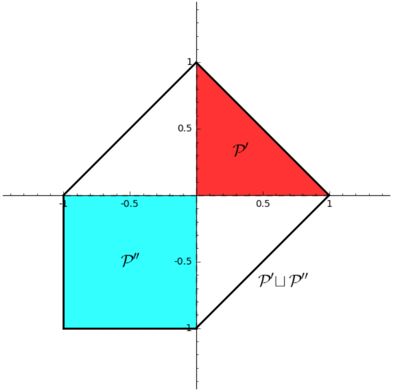
- Written by students
- Not tested as much as critical programs

Thus, we wish to **certify the analyzer**.
It requires **certified operators**.

Correction of polyhedral operators

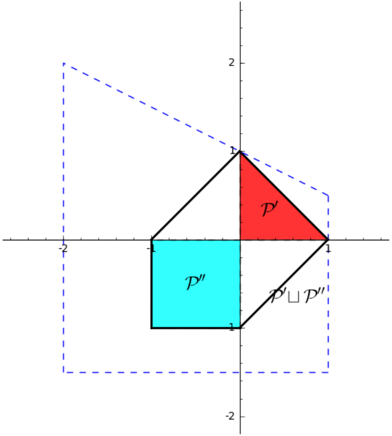
- In abstract interpretation, we over-approximate reachable memory states
- An operator is correct if its actual (computed) result includes the expected (mathematical) one
- Proofs are mainly inclusion





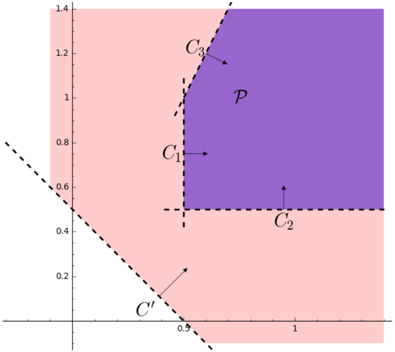
Correction of the convex hull $\mathcal{P}' \sqcup \mathcal{P}''$:

$$(\mathcal{P}' \sqsubseteq \mathcal{P}' \sqcup \mathcal{P}'') \wedge (\mathcal{P}'' \sqsubseteq \mathcal{P}' \sqcup \mathcal{P}'')$$

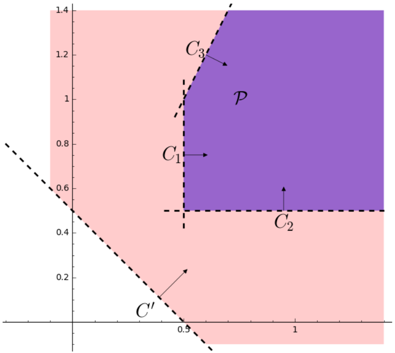


Correction of the convex hull $\mathcal{P}' \sqcup \mathcal{P}''$:

$$(\mathcal{P}' \sqsubseteq \mathcal{P}' \sqcup \mathcal{P}'') \wedge (\mathcal{P}'' \sqsubseteq \mathcal{P}' \sqcup \mathcal{P}'')$$



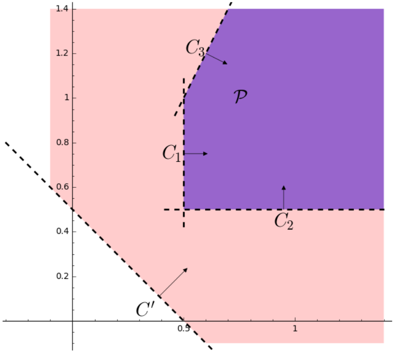
$$
\begin{array}{llcl}
\boldsymbol{C_1} & : & 2x_1 - 1 & \geq 0 \\
\boldsymbol{C_2} & : & x_2 - \frac{1}{2} & \geq 0 \\
\boldsymbol{C_3} & : & 2x_1 - x_2 & \geq 0 \\
 & & & \\
\boldsymbol{C'} & : & 2x_1 + x_2 - 1 & \geq 0
\end{array}
$$



$$\begin{array}{llr}
\boldsymbol{C_1} : & 2x_1 - 1 & \geq 0 \\
\boldsymbol{C_2} : & x_2 - \frac{1}{2} & \geq 0 \\
\boldsymbol{C_3} : & 2x_1 - x_2 & \geq 0 \\
\\
\boldsymbol{C'} : & 2x_1 + x_2 - 1 & \geq 0
\end{array}$$

Farkas' combination: $\frac{1}{2} + 1 \cdot \boldsymbol{C_1} + 1 \cdot \boldsymbol{C_2}$

$$= \frac{1}{2} + 1 \cdot \underbrace{\left(2x_1 - 1\right)}_{\boldsymbol{C_1} \geq 0} + 1 \cdot \underbrace{\left(x_2 - \frac{1}{2}\right)}_{\boldsymbol{C_2} \geq 0} = \underbrace{2x_1 + x_2 - 1}_{\boldsymbol{C'}}$$



$$\begin{array}{lcrl} C_1 & : & 2x_1 - 1 & \geq 0 \\ C_2 & : & x_2 - \frac{1}{2} & \geq 0 \\ C_3 & : & 2x_1 - x_2 & \geq 0 \end{array}$$

$$C' \; : \; 2x_1 + x_2 - 1 \geq 0$$

Farkas' combination: $\frac{1}{2} + 1 \cdot C_1 + 1 \cdot C_2$

$$= \frac{1}{2} + 1 \cdot \underbrace{\left(2x_1 - 1\right)}_{C_1 \geq 0} + 1 \cdot \underbrace{\left(x_2 - \frac{1}{2}\right)}_{C_2 \geq 0} = \underbrace{2x_1 + x_2 - 1}_{C' \geq 0}$$

The Verified Polyhedra Library (VPL) [Fouilhé et al., 2013]:

- Developed as a *relational* abstract domain in the certified analyzer Verasco [Jourdan et al., 2015]

- Polyhedral operators ($\sqsubseteq$, $\sqcup$, $\sqcap$, variable elimination) *a posteriori certified* in COQ
  - $\rightarrow$ results are computed by an OCAML program that generates Farkas' combinations
  - $\rightarrow$ simple verification of these combinations in COQ [Besson et al., 2010]

- *Constraint-only* representation of polyhedra

Double description

- Uses both representations of polyhedra
- Implemented by most polyhedra libraries
- Conversion from one representation to the other by Chernikova's algorithm

Double description

- Uses both representations of polyhedra
- Implemented by most polyhedra libraries
- Conversion from one representation to the other by Chernikova's algorithm

Unsuitable for certification:
no known way for a posteriori certification of the conversion algorithm

Double description

- Uses both representations of polyhedra
- Implemented by most polyhedra libraries
- Conversion from one representation to the other by Chernikova's algorithm

Unsuitable for certification:
no known way for a posteriori certification of the conversion algorithm

Problem: some operators ($\sqcup$, variable elimination) become expensive when restricted to constraints

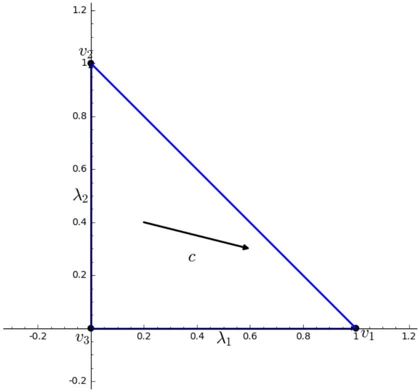
Double description

- Uses both representations of polyhedra
- Implemented by most polyhedra libraries
- Conversion from one representation to the other by Chernikova's algorithm

Unsuitable for certification:
no known way for a posteriori certification of the conversion algorithm

Problem: some operators ($\sqcup$, variable elimination) become expensive when restricted to constraints

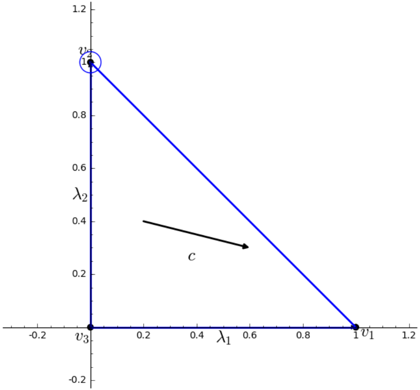
Idea: encode operators in Parametric Linear Programming



maximize *the objective* $\boldsymbol{c} \cdot (\lambda_1, \lambda_2) \triangleq 4\lambda_1 - \lambda_2$

under the constraints

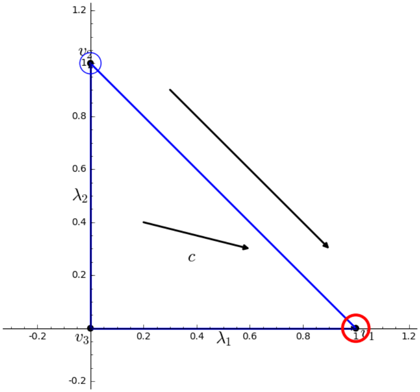
$$\lambda_1 \geq 0$$
$$\lambda_2 \geq 0$$
$$\lambda_1 + \lambda_2 \leq 1$$



maximize *the objective* $\boldsymbol{c} \cdot (\lambda_1, \lambda_2) \triangleq 4\lambda_1 - \lambda_2$

under the constraints

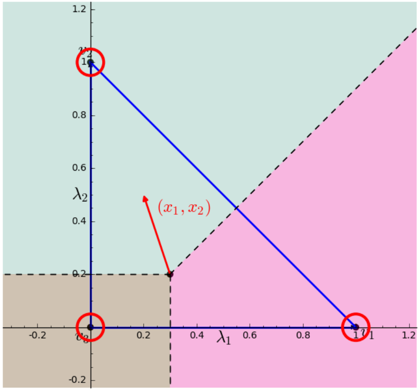
$$\lambda_1 \geq 0$$
$$\lambda_2 \geq 0$$
$$\lambda_1 + \lambda_2 \leq 1$$



maximize *the objective* $\boldsymbol{c} \cdot (\lambda_1, \lambda_2) \triangleq 4\lambda_1 - \lambda_2$

under the constraints

$$\lambda_1 \geq 0$$
$$\lambda_2 \geq 0$$
$$\lambda_1 + \lambda_2 \leq 1$$



maximize *the objective* $\boldsymbol{c}(x_1, x_2) \cdot (\lambda_1, \lambda_2) \triangleq x_1\lambda_1 + x_2\lambda_2$

under the constraints

$$\lambda_1 \geq 0$$
$$\lambda_2 \geq 0$$
$$\lambda_1 + \lambda_2 \leq 1$$

Linear Programming	Parametric Linear Programming
minimize *the objective* z: $c_0 + c_1\lambda_1 + \ldots + c_m\lambda_m$ **under the constraints** $A\boldsymbol{\lambda} \leq b$	**minimize** *the objective* $z(\boldsymbol{x})$: $c_0(\boldsymbol{x}) + c_1(\boldsymbol{x}) \cdot \lambda_1 + \ldots + c_m(\boldsymbol{x}) \cdot \lambda_m$ **under the constraints** $A\boldsymbol{\lambda} \leq b$
$\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ are decision variables c_i's are **rational constants**	$\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ are decision variables c_i's are **affine forms of the parameters** $\boldsymbol{x} = (x_1, \ldots, x_n)$

PLP encoding of polyhedral operators

Let

$$op : polyhedra \longrightarrow polyhedra$$

be a polyhedral operator that must produce **only** over-approximations

PLP encoding of polyhedral operators

Let

$$op : polyhedra \longrightarrow polyhedra$$

be a polyhedral operator that must produce **only** over-approximations, *i.e.*

$$\forall \mathcal{P} \in polyhedra, \mathcal{P} \sqsubseteq op(\mathcal{P})$$

PLP encoding of polyhedral operators

Let

$$op : polyhedra \longrightarrow polyhedra$$

be a polyhedral operator that must produce **only** over-approximations, *i.e.*

$$\forall \mathcal{P} \in polyhedra, \mathcal{P} \sqsubseteq op(\mathcal{P})$$

Then by Farkas' lemma, each constraint $\boldsymbol{C'}$ of $op(\mathcal{P})$ can be expressed as a nonnegative affine combination of constraints of $\mathcal{P}$

PLP encoding of polyhedral operators

Let

$$op : polyhedra \longrightarrow polyhedra$$

be a polyhedral operator that must produce **only** over-approximations, *i.e.*

$$\forall \mathcal{P} \in polyhedra, \mathcal{P} \sqsubseteq op(\mathcal{P})$$

Then by Farkas' lemma, each constraint $\boldsymbol{C'}$ of $op(\mathcal{P})$ can be expressed as a nonnegative affine combination of constraints of $\mathcal{P}$, *i.e.*

$$\forall \boldsymbol{x},\ \boldsymbol{C'}(\boldsymbol{x}) = \boxed{\lambda_0 + \lambda_1 \cdot \boldsymbol{C_1}(\boldsymbol{x}) + \ldots + \lambda_p \cdot \boldsymbol{C_p}(\boldsymbol{x})}$$

PLP encoding of polyhedral operators

Let

$$op : polyhedra \longrightarrow polyhedra$$

be a polyhedral operator that must produce **only** over-approximations, *i.e.*

$$\forall \mathcal{P} \in polyhedra, \mathcal{P} \sqsubseteq op(\mathcal{P})$$

Then by Farkas' lemma, each constraint $\boldsymbol{C'}$ of $op(\mathcal{P})$ can be expressed as a nonnegative affine combination of constraints of $\mathcal{P}$, *i.e.*

$$\forall \boldsymbol{x},\ \boldsymbol{C'}(\boldsymbol{x}) = \boxed{\lambda_0 + \lambda_1 \cdot \boldsymbol{C_1}(\boldsymbol{x}) + \ldots + \lambda_p \cdot \boldsymbol{C_p}(\boldsymbol{x})}$$

Objective function of a PLP problem

PLP encoding of polyhedral operators

Generic form of the PLP encoding of a polyhedral operator:

$$\begin{gathered}\textbf{minimize } \lambda_0 + \lambda_1 \cdot \boldsymbol{C_1}(\boldsymbol{x}) + \ldots + \lambda_p \cdot \boldsymbol{C_p}(\boldsymbol{x}) \\ \textbf{under the constraints:} \\ A\boldsymbol{\lambda} \leq b \\ \boldsymbol{\lambda} \geq 0\end{gathered}$$

where

- $\boldsymbol{C_1}(\boldsymbol{x}) \geq 0, \ldots, \boldsymbol{C_p}(\boldsymbol{x}) \geq 0$ are the constraints of the input polyhedron $\mathcal{P}$
- $A\boldsymbol{\lambda} \leq b$ defines the operator

Conclusion

Parametric Linear Programming

- Generic tool (used for variable elimination, convex hull, linearization)
- Avoid redundancies in the result (thanks to a normalization constraint in the PLP encoding)

Parametric Linear Programming

- Generic tool (used for variable elimination, convex hull, linearization)
- Avoid redundancies in the result (thanks to a normalization constraint in the PLP encoding)

The VPL

- Abstract domain of polyhedra certified in COQ
- Better scaling with PLP
- Handles nonlinear constraints

Conclusion

Parametric Linear Programming

- Generic tool (used for variable elimination, convex hull, linearization)
- Avoid redundancies in the result (thanks to a normalization constraint in the PLP encoding)

The VPL

- Abstract domain of polyhedra certified in COQ
- Better scaling with PLP
- Handles nonlinear constraints

Diffusion

- Available on Github (https://github.com/VERIMAG-Polyhedra)
- Opam package
- Frama-C binding: experimental branch of EVA

Perspectives

Ongoing work

- Use the VPL in Constraint Programming
- Binding in the AbSolute solver

Certification

- Provide precision certificates

Performance

- Cartesian product of polyhedra
- Parallelization of the PLP solver

References

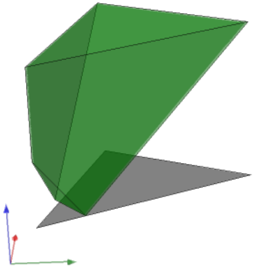
Boulmé, S. and Maréchal, A. (2015).
Refinement to certify abstract interpretations, illustrated on linearization for polyhedra.
In Interactive Theorem Proving (ITP), volume 9236 of LNCS, pages 100–116. Springer.

Boulmé, S. and Maréchal, A. (2018).
A Coq tactic for equality learning in linear arithmetic.
In Interactive Theorem Proving (ITP), LNCS. Springer.
To appear.

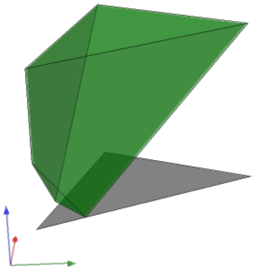
Maréchal, A., Fouilhé, A., King, T., Monniaux, D., and Périn, M. (2016).
Polyhedral approximation of multivariate polynomials using Handelman's theorem.
In Verification, Model Checking, and Abstract Interpretation (VMCAI), volume 9583 of LNCS, pages 166–184. Springer.

Maréchal, A., Monniaux, D., and Périn, M. (2017).
Scalable minimizing-operators on polyhedra via parametric linear programming.
In Static Analysis Symposium (SAS), volume 10422 of LNCS, pages 212–231. Springer.

Maréchal, A. and Périn, M. (2017).
Efficient elimination of redundancies in polyhedra by raytracing.
In Verification, Model Checking, and Abstract Interpretation (VMCAI), volume 10145 of LNCS, pages 367–385. Springer.



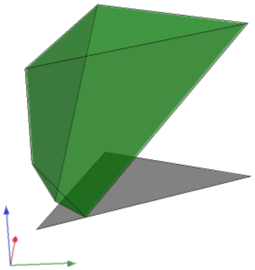
$$
\begin{array}{lcrrrcr}
\boldsymbol{C_1} & : & -x_1 & -2x_2 & +2x_3 & \geq & -7 \\
\boldsymbol{C_2} & : & -x_1 & +2x_2 & & \geq & 1 \\
\boldsymbol{C_3} & : & 3x_1 & -x_2 & & \geq & 0 \\
\boldsymbol{C_4} & : & & & -x_3 & \geq & -10 \\
\boldsymbol{C_5} & : & x_1 & +x_2 & +x_3 & \geq & 5
\end{array}
$$



$$
\begin{array}{lcrrrcr}
\boldsymbol{C_1} & : & -x_1 & -2x_2 & +2x_3 & \geq & -7 \\
\boldsymbol{C_2} & : & -x_1 & +2x_2 & & \geq & 1 \\
\boldsymbol{C_3} & : & 3x_1 & -x_2 & & \geq & 0 \\
\boldsymbol{C_4} & : & & & -x_3 & \geq & -10 \\
\boldsymbol{C_5} & : & x_1 & +x_2 & +x_3 & \geq & 5
\end{array}
$$

Fourier-Motzkin elimination

Kept constraints: $\boldsymbol{C_2}$; $\boldsymbol{C_3}$

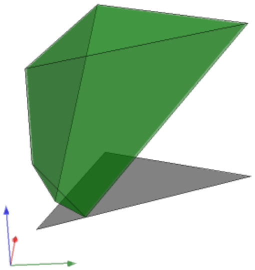


$$
\begin{array}{rcrrrcr}
\boldsymbol{C_1} & : & -x_1 & -2x_2 & +2x_3 & \geq & -7 \\
\boldsymbol{C_2} & : & -x_1 & +2x_2 & & \geq & 1 \\
\boldsymbol{C_3} & : & 3x_1 & -x_2 & & \geq & 0 \\
\boldsymbol{C_4} & : & & & -x_3 & \geq & -10 \\
\boldsymbol{C_5} & : & x_1 & +x_2 & +x_3 & \geq & 5
\end{array}
$$

Fourier-Motzkin elimination

Kept constraints: $\boldsymbol{C_2}$; $\boldsymbol{C_3}$

Combinations that cancel x_3: $\boldsymbol{C_1} + 2 \cdot \boldsymbol{C_4}$; $\boldsymbol{C_4} + \boldsymbol{C_5}$



$$\begin{array}{lcrrrcr}
\boldsymbol{C_1} & : & -x_1 & -2x_2 & +2x_3 & \geq & -7 \\
\boldsymbol{C_2} & : & -x_1 & +2x_2 & & \geq & 1 \\
\boldsymbol{C_3} & : & 3x_1 & -x_2 & & \geq & 0 \\
\boldsymbol{C_4} & : & & & -x_3 & \geq & -10 \\
\boldsymbol{C_5} & : & x_1 & +x_2 & +x_3 & \geq & 5
\end{array}$$

Fourier-Motzkin elimination

Kept constraints: $\boldsymbol{C_2}$; $\boldsymbol{C_3}$

Combinations that cancel x_3: $\boldsymbol{C_1} + 2 \cdot \boldsymbol{C_4}$; $\boldsymbol{C_4} + \boldsymbol{C_5}$

Minimization returns irredundant solutions: $\{\boldsymbol{C_2}, \boldsymbol{C_3}, \boldsymbol{C_1} + 2 \cdot \boldsymbol{C_4}\}$

$$\boldsymbol{C_4} + \boldsymbol{C_5} = \frac{29}{5} + \frac{4}{5} \cdot \boldsymbol{C_2} + \frac{3}{5} \cdot \boldsymbol{C_3}$$

Projection is an important operator used for

- variable elimination
- convex hull

Computing projection by Fourier-Motzkin elimination:

$\rightarrow$ eliminates one variable at a time;

$\rightarrow$ generates many redundant constraints (the number of constraints can double at each elimination);

$\rightarrow$ has an exponential complexity in the number of eliminated variables

$\rightarrow$ "non-geometrical" algorithm: the order of elimination has a great impact on execution time

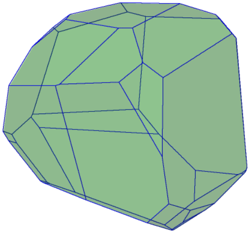
Projection problems randomly generated from:

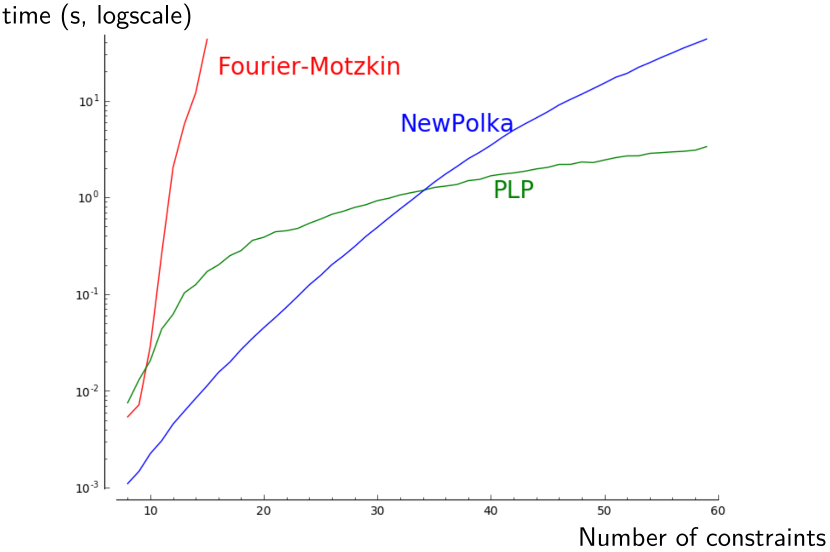
number of constraints

number of variables

density

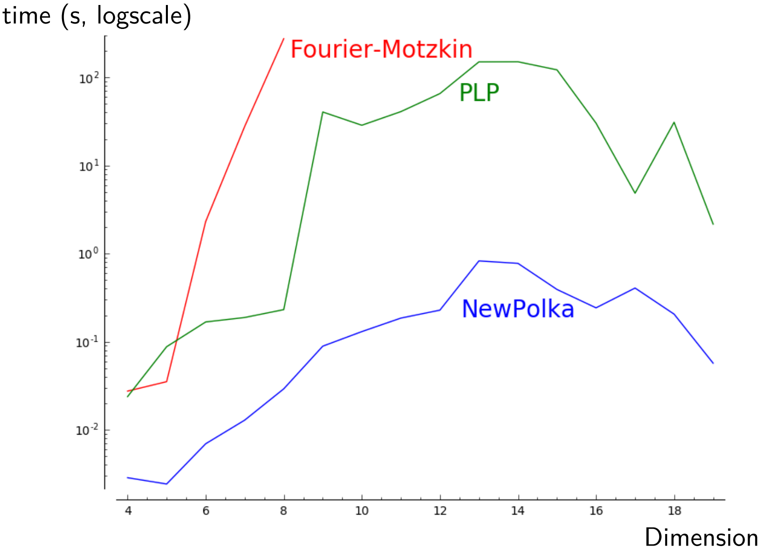
number of variables to eliminate





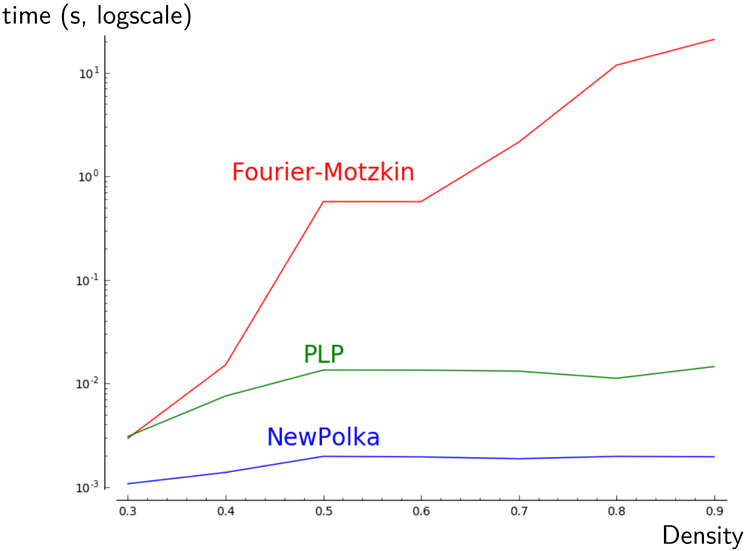
PLP scales better on big problems

Experiments on Projection via PLP



Fourier-Motzkin is still interesting for a small dimension

Experiments on Projection via PLP



PLP is less sensitive to density