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
Convex polyhedra
Where to find some?

// x ∈ [0, 5]
// y ∈ [−5, 5]

int z;
if (2*x + 8*y >= 11) {
    z = 1/y;
}

Can y be 0?

0 ≤ x ≤ 5 ∧ −5 ≤ y ≤ 5 ∧ 2x + 8y ≥ 11 ∧ y = 0

Is this feasible?
Convex polyhedra
Where to find some?

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

Can y be 0?
```
Convex polyhedra
Where to find some?

// x ∈ [0, 5]
// y ∈ [−5, 5]

int z;
if (2*x + 8*y >= 11){
    z = 1/y;
}

Can y be 0?

0 ≤ x ≤ 5
∧ −5 ≤ y ≤ 5
∧ 2x + 8y ≥ 11
∧ y = 0

Is this feasible?
Convex polyhedra
Where to find some?

// x ∈ [0, 5]
// y ∈ [−5, 5]

int z;
if (2*x + 8*y >= 11){
  z = 1/y;
}

Can y be 0?

0 ≤ x ≤ 5
∧ −5 ≤ y ≤ 5
∧ 2x + 8y ≥ 11
∧ y = 0

Is this feasible?
Convex polyhedra
Where to find some?

// x ∈ [0, 5]
// y ∈ [−5, 5]

int z;
if (2 * x + 8 * y >= 11) {
    z = 1 / y;
}

Can y be 0?

0 ≤ x ≤ 5
∧ −5 ≤ y ≤ 5
∧ 2x + 8y ≥ 11
∧ y = 0

Is this feasible?
Convex polyhedra
Where to find some?

// x ∈ [0, 5]
// y ∈ [−5, 5]

int z;
if (2∗x + 8∗y >= 11){
    z = 1/y;
}

Can y be 0?

0 ≤ x ≤ 5
∧  −5 ≤ y ≤ 5
∧  2x + 8y ≥ 11
∧  y = 0

Is this feasible?
Convex polyhedra
Two representations: constraints & generators

As constraints

\[ C_1 : 2x_1 \geq 1 \]
\[ C_2 : 2x_2 \geq 1 \]
\[ C_3 : 2x_1 - x_2 \geq 0 \]
Convex polyhedra
Two representations: constraints & generators

As *generators*: vertices and rays

\[ v_1 : \left( \frac{1}{2}, 1 \right) \]
\[ v_2 : \left( \frac{1}{2}, \frac{1}{2} \right) \]
\[ r_1 : (1, 0) \]
\[ r_2 : (1, 1) \]
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
Polyhedral operators
Meet: intersection

\[
\begin{align*}
\ldots
/* (x_1, x_2) \in P' */ \\
\text{if } (x_1 + x_2 \leq 2) \{ \\
/* (x_1, x_2) \in P' \cap \{x_1 + x_2 \leq 2\} */ \\
\}
\end{align*}
\]

\begin{align*}
P' & : \begin{cases} 
C_1: 2x_1 \geq 1 \\
C_2: 2x_2 \geq 1 \\
C_3: 2x_1 - x_2 \geq 0 
\end{cases} \\
P' \cap P'' & : \\
P'' & : \begin{cases} 
C': x_1 + x_2 \leq 2 
\end{cases}
\end{align*}
Polyhedral operators
Join: convex hull

\dots

\texttt{/*}
\texttt{if (\ldots) \{}
\texttt{\quad \ldots}
\texttt{\quad (x_1, x_2) \in P'}\texttt{ */}
\texttt{\}
\texttt{else \{}
\texttt{\quad \ldots}
\texttt{\quad (x_1, x_2) \in P'' \ */}
\texttt{\}}
\texttt{/* (x_1, x_2) \in P' \sqcup P'' */}
Certification
Why?

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
Certification

Why?

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 $\mathcal{P}' \sqcup \mathcal{P}''$:

$$(\mathcal{P}' \subseteq \mathcal{P}' \sqcup \mathcal{P}'') \land (\mathcal{P}'' \subseteq \mathcal{P}' \sqcup \mathcal{P}'')$$
Correction of the convex hull $\mathcal{P}' \cup \mathcal{P}''$:

$$(\mathcal{P}' \subseteq \mathcal{P}' \cup \mathcal{P}'') \land (\mathcal{P}'' \subseteq \mathcal{P}' \cup \mathcal{P}'')$$
How to prove a polyhedral inclusion?

Farkas’ combination: Example

\[ C_1 : \quad 2x_1 - 1 \geq 0 \]
\[ C_2 : \quad x_2 - \frac{1}{2} \geq 0 \]
\[ C_3 : \quad 2x_1 - x_2 \geq 0 \]
\[ C' : \quad 2x_1 + x_2 - 1 \geq 0 \]
How to prove a polyhedral inclusion?

**Farkas’ combination: Example**

\[ \begin{align*}
  C_1 : & \quad 2x_1 - 1 \geq 0 \\
  C_2 : & \quad x_2 - \frac{1}{2} \geq 0 \\
  C_3 : & \quad 2x_1 - x_2 \geq 0 \\
  C' : & \quad 2x_1 + x_2 - 1 \geq 0
\end{align*} \]

**Farkas’ combination:** \( \frac{1}{2} + 1 \cdot C_1 + 1 \cdot C_2 \)

\[
= \frac{1}{2} + 1 \cdot (2x_1 - 1) + 1 \cdot \left( x_2 - \frac{1}{2} \right)
\leq \begin{array}{l}
C_1 \geq 0 \\
C_2 \geq 0 \\
C' \geq 0
\end{array}
= 2x_1 + x_2 - 1 \]
How to prove a polyhedral inclusion?

Farkas’ combination: Example

\[ C_1 : \quad 2x_1 - 1 \geq 0 \]
\[ C_2 : \quad x_2 - \frac{1}{2} \geq 0 \]
\[ C_3 : \quad 2x_1 - x_2 \geq 0 \]
\[ C' : \quad 2x_1 + x_2 - 1 \geq 0 \]

Farkas’ combination: \( \frac{1}{2} + 1 \cdot C_1 + 1 \cdot C_2 \)

\[
\begin{align*}
\frac{1}{2} + 1 \cdot (2x_1 - 1) + 1 \cdot \left( x_2 - \frac{1}{2} \right) &= 2x_1 + x_2 - 1 \\
\underbrace{C_1 \geq 0}_{\text{constraint}} \quad \underbrace{C_2 \geq 0}_{\text{constraint}} \quad \underbrace{C' \geq 0}_{\text{constraint}}
\end{align*}
\]
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 (□, □, ⊓, variable elimination) *a posteriori certified* in COQ
  - results are computed by an OCAML program that generates Farkas’ combinations
  - simple verification of these combinations in COQ [Besson et al., 2010]

- *Constraint-only* representation of polyhedra
The Verified Polyhedra Library
Why constraint-only?

**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 (⊔, variable elimination) become expensive when restricted to constraints

Idea: encode operators in Parametric Linear Programming
maximize the objective $c \cdot (\lambda_1, \lambda_2) \triangleq 4\lambda_1 - \lambda_2$
under the constraints

- $\lambda_1 \geq 0$
- $\lambda_2 \geq 0$
- $\lambda_1 + \lambda_2 \leq 1$
maximize the objective $c \cdot (\lambda_1, \lambda_2) \triangleq 4\lambda_1 - \lambda_2$
under the constraints

$\lambda_1 \geq 0$
$\lambda_2 \geq 0$
$\lambda_1 + \lambda_2 \leq 1$
maximize the objective \( c \cdot (\lambda_1, \lambda_2) \triangleq 4\lambda_1 - \lambda_2 \)

under the constraints
\[
\lambda_1 \geq 0 \\
\lambda_2 \geq 0 \\
\lambda_1 + \lambda_2 \leq 1
\]
Parametric Linear Programming (PLP)

maximize the objective \( \mathbf{c} \cdot \mathbf{\lambda} \triangleq x_1 \lambda_1 + x_2 \lambda_2 \)

under the constraints

\[
\begin{align*}
\lambda_1 & \geq 0 \\
\lambda_2 & \geq 0 \\
\lambda_1 + \lambda_2 & \leq 1
\end{align*}
\]
### Linear Programming

**minimize the objective** $z$:  
\[ c_0 + c_1 \lambda_1 + \ldots + c_m \lambda_m \]

**under the constraints**  
\[ A\lambda \leq b \]

**\lambda** = $(\lambda_1, \ldots, \lambda_m)$  
are decision variables  
$c_i$’s are rational constants

---

### Parametric Linear Programming

**minimize the objective** $z(x)$:  
\[ c_0(x) + c_1(x) \cdot \lambda_1 + \ldots + c_m(x) \cdot \lambda_m \]

**under the constraints**  
\[ A\lambda \leq b \]

**\lambda** = $(\lambda_1, \ldots, \lambda_m)$  
are decision variables  
$c_i$’s are affine forms of the parameters $x = (x_1, \ldots, x_n)$
Let

$$\text{op} : \text{polyhedra} \rightarrow \text{polyhedra}$$

be a polyhedral operator that must produce **only** over-approximations
Let\[ op : \text{polyhedra} \rightarrow \text{polyhedra} \]
be a polyhedral operator that must produce \textbf{only} over-approximations, \textit{i.e.}\[
\forall P \in \text{polyhedra}, P \subseteq op(P)
\]
Let

\[
op : \text{polyhedra} \rightarrow \text{polyhedra}
\]

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

\[
\forall \mathcal{P} \in \text{polyhedra}, \mathcal{P} \subseteq \text{op}(\mathcal{P})
\]

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

\[ op : \text{polyhedra} \rightarrow \text{polyhedra} \]

be a polyhedral operator that must produce \textbf{only} over-approximations, \textit{i.e.}

\[ \forall P \in \text{polyhedra}, \ P \subseteq \text{op}(\mathcal{P}) \]

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

\[ \forall x, \ C'(x) = \left( \lambda_0 + \lambda_1 \cdot C_1(x) + \ldots + \lambda_p \cdot C_p(x) \right) \]
Let

\[ \text{op} : \text{polyhedra} \rightarrow \text{polyhedra} \]

be a polyhedral operator that must produce \textit{only} over-approximations, \textit{i.e.}

\[ \forall \mathcal{P} \in \text{polyhedra}, \mathcal{P} \subseteq \text{op}(\mathcal{P}) \]

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

\[ \forall \mathbf{x}, \ C'(\mathbf{x}) = \underbrace{\lambda_0 + \lambda_1 \cdot C_1(\mathbf{x}) + \ldots + \lambda_p \cdot C_p(\mathbf{x})}_{\text{Objective function of a PLP problem}} \]
Generic form of the PLP encoding of a polyhedral operator:

\[ \text{minimize} \quad \lambda_0 + \lambda_1 \cdot C_1(x) + \ldots + \lambda_p \cdot C_p(x) \]

under the constraints:

\[ A\lambda \leq b \]
\[ \lambda \geq 0 \]

where

- \( C_1(x) \geq 0, \ldots, C_p(x) \geq 0 \) are the constraints of the input polyhedron \( P \)
- \( A\lambda \leq b \) 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)
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
<table>
<thead>
<tr>
<th>Conclusion</th>
</tr>
</thead>
</table>

### 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
Ongoing work
- Use the $\text{VPL}$ in Constraint Programming
- Binding in the AbSolute solver

Certification
- Provide precision certificates

Performance
- Cartesian product of polyhedra
- Parallelization of the PLP solver
References


Projection with Fourier-Motzkin elimination

Example: elimination of $x_3$

\begin{align*}
C_1 : & \quad -x_1 \quad -2x_2 \quad +2x_3 \geq -7 \\
C_2 : & \quad -x_1 \quad +2x_2 \geq 1 \\
C_3 : & \quad 3x_1 \quad -x_2 \geq 0 \\
C_4 : & \quad -x_3 \geq -10 \\
C_5 : & \quad x_1 \quad +x_2 \quad +x_3 \geq 5
\end{align*}
Projection with Fourier-Motzkin elimination

Example: elimination of $x_3$

Fourier-Motzkin elimination

Kept constraints: $C_2$ ; $C_3$

\[
\begin{align*}
C_1 &: \ -x_1 \ -2x_2 \ +2x_3 \ \geq \ -7 \\
C_2 &: \ -x_1 \ +2x_2 \ \geq \ 1 \\
C_3 &: \ 3x_1 \ -x_2 \ \geq \ 0 \\
C_4 &: \ -x_3 \ \geq \ -10 \\
C_5 &: \ x_1 \ +x_2 \ +x_3 \ \geq \ 5
\end{align*}
\]
Projection with Fourier-Motzkin elimination

Example: elimination of $x_3$

\[
\begin{align*}
C_1 & : -x_1 - 2x_2 + 2x_3 \geq -7 \\
C_2 & : -x_1 + 2x_2 \geq 1 \\
C_3 & : 3x_1 - x_2 \geq 0 \\
C_4 & : -x_3 \geq -10 \\
C_5 & : x_1 + x_2 + x_3 \geq 5
\end{align*}
\]

Fourier-Motzkin elimination

Kept constraints: $C_2$ ; $C_3$

Combinations that cancel $x_3$: $C_1 + 2 \cdot C_4$ ; $C_4 + C_5$
Projection with Fourier-Motzkin elimination

Example: elimination of $x_3$

Kept constraints: $C_2$ ; $C_3$

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

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

$$C_4 + C_5 = \frac{29}{5} + \frac{4}{5} \cdot C_2 + \frac{3}{5} \cdot C_3$$
Projection with Fourier-Motzkin elimination

Limitations

Projection is an important operator used for

- variable elimination
- convex hull

Computing projection by Fourier-Motzkin elimination:

→ eliminates one variable at a time;

→ generates many redundant constraints (the number of constraints can double at each elimination);

→ has an exponential complexity in the number of eliminated variables

→ “non-geometrical” algorithm: the order of elimination has a great impact on execution time
Experiments on Projection via PLP

Projection problems randomly generated from:

- number of constraints
- number of variables
- density
- number of variables to eliminate
Experiments on Projection via PLP

PLP scales better on big problems

Number of constraints

PLP

NewPolka

Fourier-Motzkin
Fourier-Motzkin is still interesting for a small dimension.
Experiments on Projection via PLP

PLP is less sensitive to density