Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

The development of the Verimag Verified Polyhedra Library

made of untrusted parts mixing Ocaml, C++, threads, ...

and just a little bit of Coq for certification

joint work, started in 2012, with

Sylvain Boulmé, Alexis Fouilhé, Alexandre Maréchal, David Monniaux, **Michaël Périn**, Hang Yu

Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, France

Verimag
Verified
Polyhedra
Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

- OCAML implementation of standard **polyhedral operators** of a **relational abstract domain** ($\sqsubseteq$, $\sqcup$, $\sqcap$, elimination of variables, ...)

The Verimag Verified Polyhedra Library (Features)

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

- OCAML implementation of standard **polyhedral operators** of a **relational abstract domain** ($\sqsubseteq$, $\sqcup$, $\sqcap$, elimination of variables, ...)
- certifying library by **a posteriori verification** of each computation
  1. OCAML operators generate witnesses
  2. witness are checked by a simple COQ checker

The Verimag Verified Polyhedra Library (Features)

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

- OCAML implementation of standard **polyhedral operators** of a **relational abstract domain** ($\sqsubseteq$, $\sqcup$, $\sqcap$, elimination of variables, ...)
- certifying library by **a posteriori verification** of each computation
  1. OCAML operators generate witnesses
  2. witness are checked by a simple COQ checker
- **developed for the COQ-certified static analyzer** VERASCO [Jourdan+, POPL'2015], a companion tool for COMPCERT C compiler [Leroy, JAR 2009]

The Verimag Verified Polyhedra Library (Features)

Verimag Verified Polyhedra Library
Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

- OCAML implementation of standard **polyhedral operators** of a **relational abstract domain** ($\sqsubseteq$, $\sqcup$, $\sqcap$, elimination of variables, ...)
- certifying library by **a posteriori verification** of each computation
  1. OCAML operators generate witnesses
  2. witness are checked by a simple COQ checker
- **developed for the COQ-certified static analyzer** VERASCO [Jourdan+, POPL'2015], a companion tool for COMPCERT C compiler [Leroy, JAR 2009]
- can be used as **a standalone** OCAML **library**, *e.g.* in the FRAMAC static analyzer [Buhler+VMCAI'2017]

The Verimag Verified Polyhedra Library (Features)

- OCAML implementation of standard **polyhedral operators** of a **relational abstract domain** ($\sqsubseteq$, $\sqcup$, $\sqcap$, elimination of variables, ...)
- certifying library by **a posteriori verification** of each computation
  1. OCAML operators generate witnesses
  2. witness are checked by a simple COQ checker
- **developed for the COQ-certified static analyzer** VERASCO [Jourdan+, POPL'2015], a companion tool for COMPCERT C compiler [Leroy, JAR 2009]
- can be used as **a standalone** OCAML **library**, *e.g.* in the FRAMAC static analyzer [Buhler+VMCAI'2017]
- used in a new COQ-**tactic for simplifying affine expressions (S.Boulmé, A.Maréchal)** (submitted)

The Verimag Verified Polyhedra Library (Contributions)

- polyhedra library in **constraint-only** representation
- **new algorithms**
  - precise polyhedral approximation of polynomial guards
  - minimization by raytracing
  - projection via Parametric Linear Programming
- **novel certification approach using factories**
  correct by construction (by A.Maréchal, S.Boulmé)
- **efficiency issues:** parallelization, floating-point computations, external libraries (GLPK,GMP,EIGEN,FLINT), reconstruction of the exact solution (on $\mathbb{Q}$)
- **available at** `github.com/VERIMAG-Polyhedra`
- state of the art **Parametric Linear Programming Solver**

Topics

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

- Polyhedra: basics
- All you need to understand the field of Polyhedra
  Farkas combinations and Linear Programming
- Why certifying software verification tools?
- Certification by result verification
- ... with as little COQ as possible
- Why another polyhedra library?
- Why are polyhedra expensive?
- Revisiting the algorithmic
- Experimental results
- Will Polyhedra be usable?

Convex Polyhedra

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

capture **affine relations between program variables** such as

inequalities: $x_1 - 2x_2 \geq 3x_3 \rightsquigarrow x_1 - 2x_2 - 3x_3 \geq 0$

boundaries: $2 \leq x_1 \leq 3 \rightsquigarrow x_1 - 2 \geq 0 \bigwedge -x_1 - 3 \geq 0$

equalities: $x_1 = x_2 + 2 \rightsquigarrow \left\{ \begin{array}{l} x_1 - x_2 - 2 \geq 0 \\ x_2 - x_1 + 2 \geq 0 \end{array} \right\}$

affine form = linear form + constant

Definition

*A **convex polyhedron** is a set of vectors* $(x_1, ..., x_n) \in \mathbb{Q}^n$ *satisfying*
a system of affine inequalities between variables $x_1, ..., x_n$

Remark (It is convex)

if two points are in the set, the segment also is.

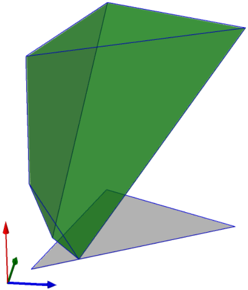
A 3D polyhedron ...

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

... as system of constraints

$$\left\{ \begin{array}{lllll} \boldsymbol{C_1} : & x_1 + 2x_2 - 2x_3 & \leq & 7, \\ \boldsymbol{C_2} : & x_1 - 2x_2 & \leq & -1, \\ \boldsymbol{C_3} : & -3x_1 + x_2 & \leq & 0, \\ \boldsymbol{C_4} : & x_3 & \leq & 10, \\ \boldsymbol{C_5} : & -x_1 - x_2 - x_3 & \leq & -5 \end{array} \right\}$$



Uses of Polyhedra

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

- **Linear Programming**
  - optimization of a cost function under affine inequalities,
  - decide the existence of a solution fulfilling affine inequalities

Uses of Polyhedra

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

- **Linear Programming**
  - optimization of a cost function under affine inequalities,
  - decide the existence of a solution fulfilling affine inequalities
- **Loop Optimization**
  "The Polyhedron Model" [Feautrier and Lengauer, 2011]
  1. approximate by a polyhedron the cells of a n-dimensions array to be updated by a loop
  2. compute vectors that exactly describe that space of cells
  3. generate the optimized loop

Uses of Polyhedra

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

- **Linear Programming**
  - optimization of a cost function under affine inequalities,
  - decide the existence of a solution fulfilling affine inequalities
- **Loop Optimization**
  "The Polyhedron Model" [Feautrier and Lengauer, 2011]
  1. approximate by a polyhedron the cells of a n-dimensions array to be updated by a loop
  2. compute vectors that exactly describe that space of cells
  3. generate the optimized loop
- **Static Analysis of Programs**
  POPL'78 [Cousot and Halbwachs, 1978]
  - capture affine relation between variables
  - discover implicit equalities
  - more precise than interval analysis but costlier

Why certifying results of Software Verification Tools?

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

An old greek syllogism

Programs contain bugs.

Software Verification Tools are programs.

Thus, ___s contain ___s

Why certifying results of Software Verification Tools?

Verimag **V**erified **P**olyhedra **L**ibrary

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

An old greek syllogism

Programs contain bugs.
Software Verification Tools are programs.
Thus, <u>S</u><u>V</u><u>T</u>s contain <u>B</u><u>U</u><u>G</u>s

Why certifying results of Software Verification Tools?

Verimag **V**erified **P**olyhedra **L**ibrary

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

An old greek syllogism

Programs contain bugs.
Software Verification Tools are programs.
Thus, SVTs contain BUGs

… more than other programs

- mostly prototypes developed by several students
- complex underlying theory
- less users, less tested

Three ways to gain confidence in SVT

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

1. **No tool is trustable ... but if they agree on the result**
   - Running each tool with (same inputs $\rightsquigarrow$ same answer) increases the confidence
   - **Quantifiable?** How **many tools** to reach P.Failure$=10^{-9}$?

Three ways to gain confidence in SVT

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

1. **No tool is trustable … but if they agree on the result**
   - Running each tool with (same inputs $\rightsquigarrow$ same answer) increases the confidence
   - **Quantifiable?** How **many tools** to reach P.Failure$=10^{-9}$?
2. **Only trust the proof-checker … which becomes critical**
   - extend SVT to **generate certificates**

     SAT, UNSAT, $\Longrightarrow_{Theory}$, $[\![\text{Program}]\!] \models$ Properties
   - $[\![\ldots]\!]$ is a formal semantics of the programming language

     (*e.g.* CompCert C semantics in COQ)
   - How long **can a bug stay silent** in the proof-checker?

     (the COQ-engine is not as simple as it used to be)

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Three ways to gain confidence in SVT

1. **No tool is trustable … but if they agree on the result**
   - Running each tool with (same inputs $\rightsquigarrow$ same answer) increases the confidence
   - **Quantifiable?** How **many tools** to reach P.Failure$=10^{-9}$?

2. **Only trust the proof-checker … which becomes critical**
   - extend SVT to **generate certificates**

     SAT, UNSAT, $\Longrightarrow_{Theory}$, $[\![\text{Program}]\!] \models$ Properties
   - $[\![\ldots]\!]$ is a formal semantics of the programming language

     (*e.g.* COMPCERT C semantics in COQ)
   - How long **can a bug stay silent** in the proof-checker?

     (the COQ-engine is not as simple as it used to be)

3. **Correct by extraction using** COQ (*e.g.* COMPCERT)

$$\underbrace{\text{proof(algo} \models \text{spec)}}_{\text{in COQ}} \xrightarrow{\text{extraction}} \text{OCAML program}$$

Certification versus Result Verification

Verimag Verified Polyhedra Library
Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Full COQ-certified development (COMPCERT, VERASCO S.A)

- time consuming (refactoring the code means adapting the proofs)
- requires proof skills
- the algorithms must be designed to be easy to prove
- **correctness of all results guaranteed** by a COQ-proof

Result verification (COMPCERT, B.S.A [Besson et al., 2010], VPL)

- external libraries (untrusted code)
- **correctness of each result is checked by** COQ
  1. external code generates witness of correctness
  2. verification by a **simple COQ-certified checker**

What must be proved for over-approximations?

- **join/union** operator $(P' := P_1 \sqcup P_2)$ is sound if $P_1 \sqsubseteq P'$ and $P_2 \sqsubseteq P'$
- **meet/intersection** operator $(P' := P_1 \sqcap P_2)$ is sound if $P' \sqsubseteq P_1$ and $P' \sqsubseteq P_2$
- **minimization** operator $(P' := \mathit{min}\ (P))$ is **sound** if $P \sqsubseteq P'$ and **precise** if $P' \sqsubseteq P$
- **elimination/projection** $(P' := \mathit{elim}\ \ \{x\}\ \ P)$ is sound if $P \sqsubseteq P'$ and x is unbounded in P'

Soundness boils down to inclusion [Besson et al., 2010]

$P_1 \sqsubseteq P_2$ can be proved by Farkas combinations

Each operator must provide **Farkas combinations** to prove the **required inclusions**

Farkas combinations I

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Example

$$\begin{aligned} P &= \left\{ \begin{matrix} \boldsymbol{C_1} : x_1 - x_2 - 1 \geq 0, \\ \boldsymbol{C_2} : x_1 + 2x_2 + 1 \geq 0 \end{matrix} \right\} \\ P' &= \{ \boldsymbol{C'} : 3x_1 - 1 \geq 0 \} \end{aligned}$$

The Farkas Combination $2 \times \boldsymbol{C_1} + 1 \times \boldsymbol{C_2}$ ... is $\boldsymbol{C'}$

$$\overbrace{2x_1 - 2x_2 - 2 \geq 0}^{2 \times \boldsymbol{C_1}} + \overbrace{x_1 + 2x_2 + 1 \geq 0}^{1 \times \boldsymbol{C_2}} = \underbrace{3x_1 - 1 \geq 0}_{\boldsymbol{C'}}$$

It shows that $\{\boldsymbol{C_1}, \boldsymbol{C_2}\} \sqsubseteq \{\boldsymbol{C'}\}$

Farkas combinations II

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Farkas combination of constraints (linear version)

$\boldsymbol{x} \in P := \{\boldsymbol{C_1}, \ldots, \boldsymbol{C_k}\}$ *means*
$\boldsymbol{x}$ *satisfies* $\boldsymbol{C_1}(\boldsymbol{x}) \geq 0 \wedge \ldots \wedge \boldsymbol{C_k}(\boldsymbol{x}) \geq 0$ *then,*
for any ***non-negative scalars*** $\lambda_1, \ldots, \lambda_k \in \mathbb{Q}$

$$\underbrace{\lambda_1 \times \boldsymbol{C_1}(\boldsymbol{x})}_{\geq 0} + \ldots + \underbrace{\lambda_k \times \boldsymbol{C_k}(\boldsymbol{x})}_{\geq 0} \geq 0$$

Farkas combinations II

Verimag
Verified
Polyhedra
Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Farkas combination of constraints (linear version)

$\boldsymbol{x} \in P := \{\boldsymbol{C_1}, \ldots, \boldsymbol{C_k}\}$ *means*
$\boldsymbol{x}$ *satisfies* $\boldsymbol{C_1}(\boldsymbol{x}) \geq 0 \wedge \ldots \wedge \boldsymbol{C_k}(\boldsymbol{x}) \geq 0$ *then,*
for any ***non-negative scalars*** $\lambda_1, \ldots, \lambda_k \in \mathbb{Q}$

$$\underbrace{\lambda_1 \times \boldsymbol{C_1}(\boldsymbol{x})}_{\geq 0} + \ldots + \underbrace{\lambda_k \times \boldsymbol{C_k}(\boldsymbol{x})}_{\geq 0} \geq 0$$

Now, if $\boldsymbol{C'} = \lambda_1 \times \boldsymbol{C_1} + \ldots + \lambda_k \times \boldsymbol{C_k}$ *then*

$$\forall \boldsymbol{x},\ \boldsymbol{C_1}(\boldsymbol{x}) \geq 0 \wedge \ldots \wedge \boldsymbol{C_k}(\boldsymbol{x}) \geq 0 \Longrightarrow \boldsymbol{C'}(\boldsymbol{x}) \geq 0$$

which is the definition of the geometric inclusion

$$\{\boldsymbol{C_1}, \ldots, \boldsymbol{C_k}\} \sqsubseteq \{\boldsymbol{C'}\}$$

Farkas combinations III

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

How to find the Farkas coefficient λ's

$$
\begin{array}{rcl}
P & = & \left\{ \begin{array}{lrrrl}
\boldsymbol{C_1}: & 1\,x_1 & -1\,x_2 & -1 & \geq 0, \\
\boldsymbol{C_2}: & 1\,x_1 & +2\,x_2 & +1 & \geq 0, \\
\boldsymbol{C_3}: & -1\,x_1 & +1\,x_2 & +1 & \geq 0
\end{array} \right\} \\
P' & = & \left\{ \begin{array}{lrrrl}
\boldsymbol{C'}: & 3\,x_1 & +0\,x_2 & -1 & \geq 0
\end{array} \right\}
\end{array}
$$

$$
\exists ? \lambda_i \geq 0 \quad \lambda_1 \times \boldsymbol{C_1} \; + \; \lambda_2 \times \boldsymbol{C_2} \; + \; \lambda_3 \times \boldsymbol{C_3} \; = \; \boldsymbol{C'}
$$

Farkas combinations III

How to find the Farkas coefficient λ's

$$
\begin{array}{rcl}
P & = & \left\{ \begin{array}{lrrrl}
\boldsymbol{C_1}: & \mathbf{1}\,x_1 & -1\,x_2 & -1 & \geq 0, \\
\boldsymbol{C_2}: & \mathbf{1}\,x_1 & +2\,x_2 & +1 & \geq 0, \\
\boldsymbol{C_3}: & \mathbf{-1}\,x_1 & +1\,x_2 & +1 & \geq 0
\end{array} \right\} \\
P' & = & \left\{ \begin{array}{lrrrl}
\boldsymbol{C'}: & \mathbf{3}\,x_1 & +0\,x_2 & -1 & \geq 0
\end{array} \right\}
\end{array}
$$

$$
\begin{array}{cccccccc}
\exists?\lambda_i \geq 0 & \lambda_1 \times \boldsymbol{C_1} & + & \lambda_2 \times \boldsymbol{C_2} & + & \lambda_3 \times \boldsymbol{C_3} & = & \boldsymbol{C'} \\
\hline\hline
(x_1) & \lambda_1 \times \mathbf{1} & + & \lambda_2 \times \mathbf{1} & + & \lambda_3 \times \mathbf{-1} & = & \mathbf{3}
\end{array}
$$

Farkas combinations III

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

How to find the Farkas coefficient λ's

$$P = \left\{ \begin{array}{lrrrl} \boldsymbol{C_1}: & 1\,x_1 & -\mathbf{1}\,x_2 & -1 & \geq 0, \\ \boldsymbol{C_2}: & 1\,x_1 & +\mathbf{2}\,x_2 & +1 & \geq 0, \\ \boldsymbol{C_3}: & -1\,x_1 & +\mathbf{1}\,x_2 & +1 & \geq 0 \end{array} \right\}$$

$$P' = \left\{ \boldsymbol{C'} : \quad 3\,x_1 \quad +\mathbf{0}\,x_2 \quad -1 \quad \geq 0 \right\}$$

$$\begin{array}{cccccccc} \exists ? \lambda_i \geq 0 & \lambda_1 \times \boldsymbol{C_1} & + & \lambda_2 \times \boldsymbol{C_2} & + & \lambda_3 \times \boldsymbol{C_3} & = & \boldsymbol{C'} \\ \hline (x_1) & \lambda_1 \times 1 & + & \lambda_2 \times 1 & + & \lambda_3 \times -1 & = & 3 \\ (x_2) & \lambda_1 \times \textbf{-1} & + & \lambda_2 \times \mathbf{2} & + & \lambda_3 \times \mathbf{1} & = & \mathbf{0} \end{array}$$

Farkas combinations III

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

How to find the Farkas coefficient λ's

$$P = \left\{ \begin{array}{lrrrl} \boldsymbol{C_1}: & 1\,x_1 & -1\,x_2 & -\mathbf{1} & \geq 0, \\ \boldsymbol{C_2}: & 1\,x_1 & +2\,x_2 & +\mathbf{1} & \geq 0, \\ \boldsymbol{C_3}: & -1\,x_1 & +1\,x_2 & +\mathbf{1} & \geq 0 \end{array} \right\}$$

$$P' = \left\{ \boldsymbol{C'} : \quad 3\,x_1 \quad +0\,x_2 \quad -\mathbf{1} \quad \geq 0 \right\}$$

$$\begin{array}{rlllll} \exists?\lambda_i \geq 0 & \lambda_1 \times \boldsymbol{C_1} & + & \lambda_2 \times \boldsymbol{C_2} & + & \lambda_3 \times \boldsymbol{C_3} & = & \boldsymbol{C'} \\ \hline\hline (x_1) & \lambda_1 \times 1 & + & \lambda_2 \times 1 & + & \lambda_3 \times -1 & = & 3 \\ (x_2) & \lambda_1 \times -1 & + & \lambda_2 \times 2 & + & \lambda_3 \times 1 & = & 0 \\ (cst) & \lambda_1 \times \textbf{-1} & + & \lambda_2 \times \mathbf{1} & + & \lambda_3 \times \mathbf{1} & = & \textbf{-1} \end{array}$$

Linear Algebra / Linear Programming

Gauss Resolution (Linear Algebra)

It gives solutions of the systems of equalities

$$\begin{array}{rl} (x_1) & \lambda_3 = -3 + \lambda_1 + \lambda_2 \\ (x_2) & \lambda_2 = 1 \\ (cst) & 0 \times \lambda_1 = 0 \quad \lambda_1 \textit{ is free, thus choose } \lambda_1 \geq 0 \end{array}$$

But some are not Farkas Combinations *i.e.* satisfying $\lambda_i \geq 0$

$$(\lambda_1, \lambda_2, \lambda_3) = \{ (0, 1, -2), (1, 1, -1), (2, 1, 0), (3, 1, 1), \ldots \}$$

$\exists ? \lambda_i \geq 0$ ... is not in the realm of Linear Algebra
but that of Linear Programming

Linear Algebra / Linear Programming

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

The Simplex Algorithm (Linear Programming)

It's a way to choose pivots in Gauss elimination

$$\begin{array}{rl} (x_1) & \lambda_1 = 3 - \lambda_2 + \lambda_3 \\ (x_2) & \lambda_2 = 1 \\ (cst) & 0 \times \lambda_3 = 0 \quad \lambda_3 \textit{ is free, thus choose } \lambda_3 \geq 0 \end{array}$$

It focuses on Farkas Combinations *i.e.* satisfying $\lambda_i \geq 0$

$$(\lambda_1, \lambda_2, \lambda_3) = \{(2,1,0), (3,1,1), (4,1,2), (5,1,3), \dots\}$$

From efficient floating-point solver to exact solutions in $\mathbb{Q}$

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Efficient floating-point simplex algorithms (such as GLPK) do not provide exact solution (due to rounding)

$$(\lambda_1, \lambda_2, \lambda_3) = (1.99...,\ 1.0,\ 0.0...1)$$

But they are trustable on **variables that must be null** (*e.g.* $\lambda_3 = 0$) from which we can use **fast linear algorithm over the rationals (FLINT)** to solve the simplified system and obtain an exact solution $(\lambda_1, \lambda_2, \lambda_3) = (2, 1, 0)$

$\exists ? \lambda_i \geq 0$	$\lambda_1 \times \boldsymbol{C_1}$	$+$	$\lambda_2 \times \boldsymbol{C_2}$	$+$	$0 \times \boldsymbol{C_3}$	$=$	$\boldsymbol{C'}$
(x_1)	$\lambda_1 \times 1$	$+$	$\lambda_2 \times 1$			$=$	3
(x_2)	$\lambda_1 \times -1$	$+$	$\lambda_2 \times 2$			$=$	0
(cst)	$\lambda_1 \times -1$	$+$	$\lambda_2 \times 1$			$=$	-1

Polyhedra inclusion checker in COQ

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

1. **Original work in 2010** [Besson et al., 2010]
   - results checking of Bytecode Static Analyzer
   - operations are performed by NewPolka [Jeannet and Miné, 2009]
   - witnesses are computed afterwards by solving Linear Programming problems
2. **VPL, started in 2012**
   - produces witnesses on-the-fly (no duplicate computation)
   - constraint-only representation

The inclusion checker (COQ code) (Definitions & Lemma)

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

Definition Polyhedra := list (cstr $\mathbb{Q}$).

Definition *sat* $(x : \text{Vec})\ (p : \text{Polyhedra}) : \text{Prop} :=$
    $\text{List.Forall}\ p\ (\text{fun}\ (c : \text{csrt}\ \mathbb{Q}) \rightarrow c(x) \geq 0).$

Definition (infix $\sqsubseteq$) $(p_1\ p_2 : \text{Polyhedra}) : \text{Prop} :=$
    $\forall x : \text{Vec},\ \textit{sat}\ \ x\ p_1 \Rightarrow \textit{sat}\ \ x\ p_2.$

Lemma *Farkas* : $\forall\ (\Lambda : \text{list (list}\ \mathbb{Q}))\ (p_1\ p_2 : \text{Polyhedra})$
    $$(\forall \lambda \in \Lambda, \lambda \geq 0)\ \wedge\ (\underbrace{\textit{combine}}_{\simeq \textit{matrix-product}}\ \Lambda\ p_1) = p_2 \implies p_1 \sqsubseteq p_2$$

The inclusion checker (COQ code) (Program & Extraction)

Definition *inclusion_checker*
  (Λ : list (list $\mathbb{Q}$)) (p_1 p_2 : Polyhedra) : option ($p_1 \sqsubseteq p_2$) :=
 let *nn* := (*non_negative* Λ) **in**
 let *eq* := (*equal* (*combine* Λ p_1) p_2)
 in match (*nn*,*eq*) **with**
  | (**Some** proof_nn, **Some** proof_eq)
   $\rightarrow$ **Some** (*Farkas* Λ p_1 p_2 proof_nn proof_eq)
  | (_,_) $\rightarrow$ **None**
 end

COQ *inclusion_checker* $: (\Lambda, p_1, p_2) \rightarrow \begin{cases} \textbf{Some}\ (p_1 \sqsubseteq p_2) \\ \textbf{None} \end{cases}$

$\downarrow$ $\boxed{\textbf{extraction}}$

OCAML *inclusion_checker* $: (\Lambda, p_1, p_2) \rightarrow$ *bool*

Using the COQ checker in OCAML code

Verimag Verified Polyhedra Library
Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Illustration on the convex-hull operator of the VPL

The convex-hull operator $P := P_1 \sqcup P_2$ is sound if

$$P_1 \sqsubseteq P \text{ and } P_2 \sqsubseteq P$$

which is proved using two Farkas inclusion witnesses Λ_1 and Λ_2 using

$$\textit{inclusion\_checker}\ (\Lambda_1, P_1, P) = \textbf{Some}(P_1 \sqsubseteq P)$$

$$\textit{inclusion\_checker}\ (\Lambda_2, P_2, P) = \textbf{Some}(P_2 \sqsubseteq P)$$

The convex-hull (OCAML code)

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work


```
type polyhedra = { ocaml: (rat cstr) list ; coq: (ℚ cstr) list }

let convex_hull  (p1:polyhedra) (p2:polyhedra) : polyhedra =
  let (f1, f2, pOcaml) =                                      } compute
       untrusted_convex_hull p1.ocaml  p2.ocaml
  in let Λ1    = rat_to_ℚ  f1                                  }
         Λ2    = rat_to_ℚ  f2                                  }
         pCoq = rat_to_ℚ  pOcaml                               }
  in  if (inclusion_checker Λ1  p1.coq  pCoq)                  } check
        && (inclusion_checker Λ2  p2.coq  pCoq)                }
     then { ocaml = pOcaml ; coq = pCoq }                      }
     else  error  "convex_hull"                                }
```

What is guaranteed by result verification?

Verimag Verified Polyhedra Library

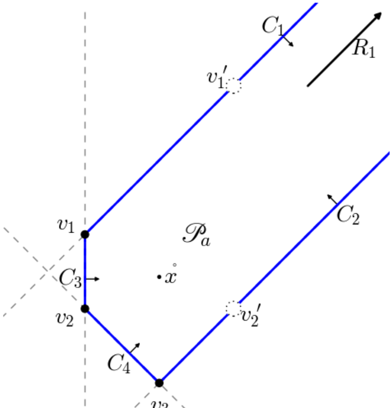
Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

- The checker is extracted in OCAML but still uses COQ representations (trustable but inefficient)
  type nat = O | S of nat
  type positive = B1 of positive | B0 of positive | BH
  e.g. 5 ≃ S(S(S(S(S(O))))) ≃ B1(B0(B1(BH)))
- **12% overhead when the COQ checker is activated** (conversion into COQ representation then computations)
- **The COQ checker can be de/activated.**
- **The equality (p.ocaml = p.coq) cannot be guaranteed**
- **Guaranty:** the COQ side mimics the computations of the untrusted side and **the COQ side checks soundness**
- S.Boulmé noticed that *"the COQ type of polyhedra can even be an opaque abstract data type or a generic type 'α"* leading to new certification means using **factories**.

The Double Description of Polyhedra

used in most Polyhedra Libraries (NewPolka, PPL, Cudd, ...)

$$\left(\begin{array}{ccccc} \textbf{as constraints} & & & \textbf{as generators} & \\ \{\boldsymbol{C_1}, \boldsymbol{C_2}, \boldsymbol{C_3}, \boldsymbol{C_4}\} & , & \{\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}\} & + & \{\boldsymbol{R_1}\} \\ & & \text{vertices} & + & \text{rays} \end{array} \right)$$



Why are polyhedra expensive ? I

Verimag Verified Polyhedra Library
Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Polyhedra as generators: $\mathscr{G} \sqcap \{ C' \}$

The intersection with one constraint can **double the number of generators**.

Example (A sliced tube unbounded in one direction)

$$\mathscr{G} = \{ \boldsymbol{v_1}, \ldots, \boldsymbol{v_n} \} + \{ \boldsymbol{r_1} \}$$

$$\mathscr{G} \sqcap \{ \boldsymbol{C'} \} = \{ \boldsymbol{v_1}, \ldots, \boldsymbol{v_n} \} + \{ \boldsymbol{v'_1}, \ldots, \boldsymbol{v'_n} \}$$

Why are polyhedra expensive ? II

Verimag **V**erified **P**olyhedra **L**ibrary

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

Polyhedra as constraints: *elim* $(x_3, \mathscr{C})$ = projection on (x_1, x_2)

The elimination of one variable can **double the number of constraints**

Example (An orange segment)

$$\mathscr{C} = \{\boldsymbol{C'}, \boldsymbol{C''}, \ \ \boldsymbol{C_1}, \ldots, \boldsymbol{C_n}\}$$

$$\textit{elim}\ (x_3, \mathscr{C}) = \{\boldsymbol{C'_1}, \ldots, \boldsymbol{C'_n}, \boldsymbol{C''_1}, \ldots, \boldsymbol{C''_n}\}$$

$$\textit{elim}\ (\{x_3, x_2\}, \mathscr{C}) = \{b \le x_1, \ x_1 \le b'\}$$

Choosing the good representation $\mathscr{C}$? $\mathscr{G}$?

Verimag
Verified
Polyhedra
Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

- The polyhedra representation can double on basic operations ($\sqcap$, *elim*)
- Sequential eliminations of variables is exponential on constraints *elim* $[x_1 \ ; \ \ldots \ ; \ x_n]$ $\mathscr{C}$

  based on Fourier-Motzkin's elimination of **one** variable
- sequential intersections is exponential on generators

$$\mathscr{G} \sqcap [\boldsymbol{C_1} \ ; \ \ldots \ ; \ \boldsymbol{C_k}]$$

  based on Chernikova's intersection with **one** cutting plane
- $\mathscr{C}$ is needed for intersection and widening, used for inclusion and minimization
- DD can choose the best algorithm or an even better algorithm using $(\mathscr{C}, \mathscr{G})$

Why a polyhedra library in constraint-only?

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

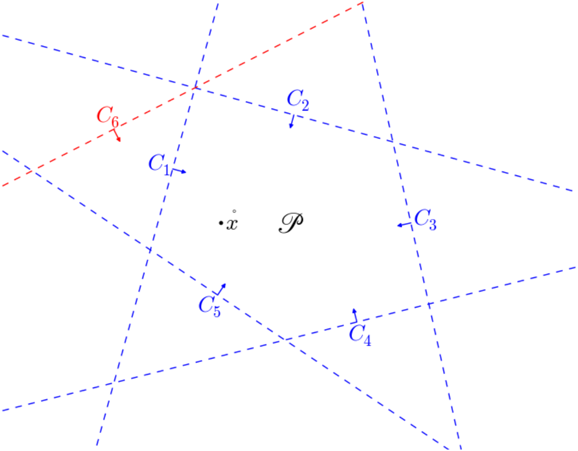
Conclusion
Related work

- **no polynomial algorithm to check equivalence** $(\mathscr{C}, \mathscr{G})$ (it probably does not exists)
- DD would have meant
  - **implementing** a naive version of **Chernikova**'s conversion algorithm **in COQ**,
  - proving it correct then
  - extracting to OCAML to get a correct but inefficient algorithm
- out of curiosity: no conversion, less memory space, **can it be as efficient as DD?**
- could we do better than **Fourier-Motzkin one-variable elimination** algorithm?

  [Howe and King, 2012]: Parametric Linear Programming can elimate several variables simultaneously
- could we improve **minimization**?

Minimization by Raytracing

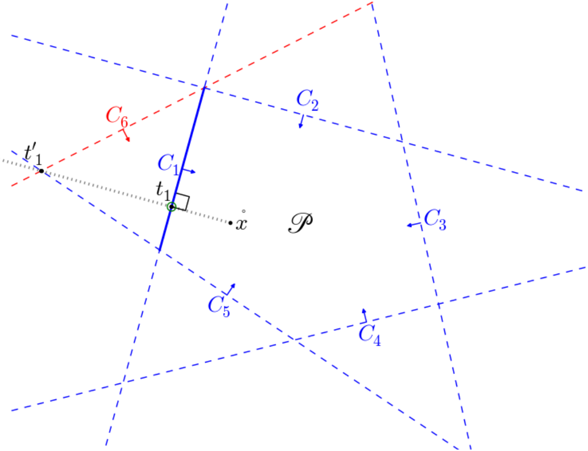
Launch a ray orthogonally to each constraint



the first constraint hit by the ray is irredundant

Minimization by Raytracing

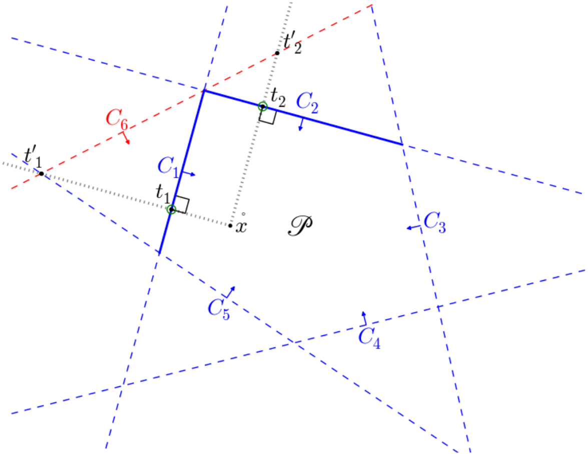
Launch a ray orthogonally to each constraint



the first constraint hit by the ray is irredundant

Minimization by Raytracing

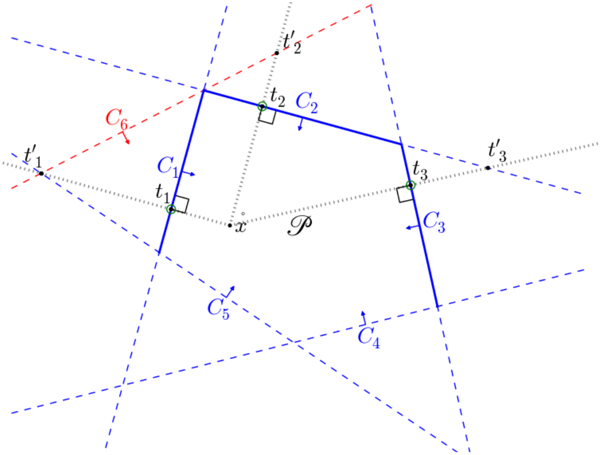
Launch a ray orthogonally to each constraint



the first constraint hit by the ray is irredundant

Minimization by Raytracing

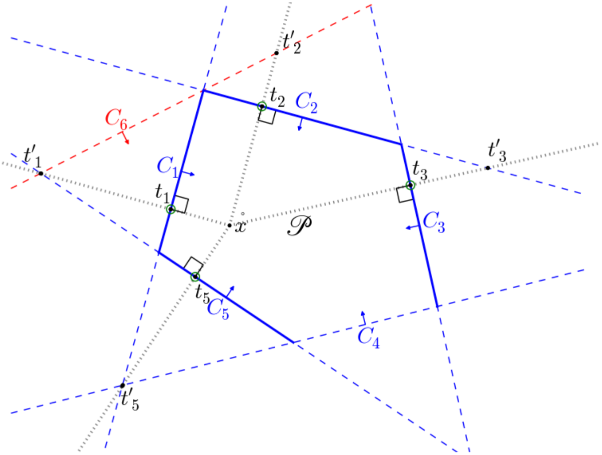
Launch a ray orthogonally to each constraint



the first constraint hit by the ray is irredundant

Minimization by Raytracing

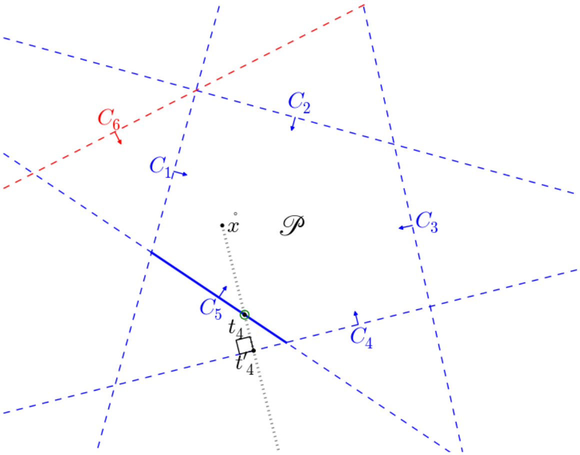
Launch a ray orthogonally to each constraint



the first constraint hit by the ray is irredundant

Minimization by Raytracing

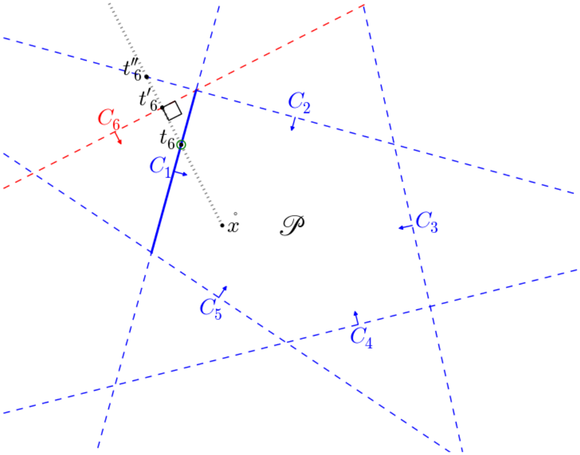
Launch a ray orthogonally to each constraint



the first constraint hit by the ray is irredundant

Minimization by Raytracing

Launch a ray orthogonally to each constraint



the first constraint hit by the ray is irredundant

Verimag Verified Polyhedra Library
Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

When raytracing fails

We use the Standard Minimization Algorithm
= inclusion testing
= existence of a Farkas Combination

- $\{\boldsymbol{C_3}, \boldsymbol{C_5}\} \subseteq \{\boldsymbol{C_4}\}$? yes
- $\{\boldsymbol{C_1}, \boldsymbol{C_2}\} \subseteq \{\boldsymbol{C_6}\}$? no

Finally,
$\boldsymbol{C_1}, \boldsymbol{C_2}, \boldsymbol{C_3}, \boldsymbol{C_5}$ were determined without solving LP problems,
it only costs a matrix-matrix product:
matrix of constraints $\times$ matrix of rays

Experiments

We compared three algorithms:

1. The standard algorithm (SMA)
   Detecting redundancies by finding Farkas combinations.
   Requires one Linear Programming for each constraint.
   Each Linear Programming contains all the constraints.
2. Raytracing with rationals (RRT)
   using a rational simplex for finding Farkas combination
3. Raytracing with floating points (FRT)
   using a floating simplex (GLPK) then reconstruction

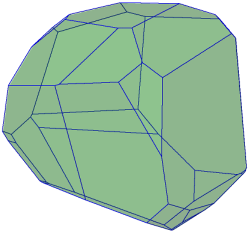
Experiments on random polyhedra

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work

We generated random polyhedra to study the sensitivity of algorithms to

- the number of variables
- the number of constraints
- the number of generators
- the redundancy rate
- the density



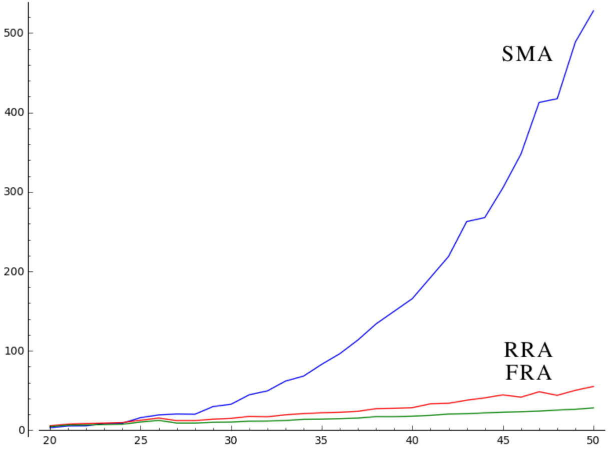
Example (6 variables, density 50%)

$0\,x_1 + 21x_2 + 0\,x_3 + 26x_4 + 0\,x_5 - 13x_6 \leq 20$

Experimental results

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra
Certification
Why ? How ?
VPL correctness
Farkas Lemma
The certified checker
Certification in COQ
Usage in VPL
Computation
Representation of Polyhedra
New algorithms
Experiments
Ongoing Work
Conclusion
Related work



Time (ms) when varying then **number of constraints**, with 10 variables, a redundancy rate of 50%, and a density of 50%.

Comparison with Chernikova's conversion algorithm

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

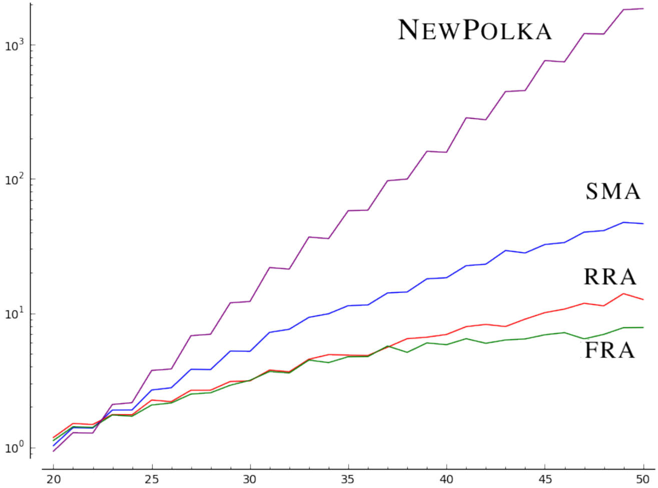
Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work



Time (ms) in log scale

Ongoing Work I (Alexandre MARÉCHAL)

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

Experimentation is not easy

- **Comparing libraries: How to be fair?**
  - DD delay some operations (conversion, minimization)
  - start by a conversion to build $(\mathscr{C}, \mathscr{G})$,
  - n-dimenions hypercubes (2^n generators) kill them
- **Using which analyzer?**

  static analyzers have been designed for intervals, not ofr polyhedra (Frama-C, VERASCO)
  - too much (useless) variables involved
  - duplication of computations
  - $(P \sqcup P') = P'$ instead of $P \sqsubseteq P'$

Ongoing Work I (Alexandre MARÉCHAL)

An experimental setup is under development

Goal: **Profile each operator** on random polyhedra **to study sensitivity** to the number of variables, of constraints, of generators, of redundancies, and the density (number of nonzero coefficients)

- record the call to polyhedral operators during analysis of
- realistic programs with a polyhedra-aware static analyzer
- extract significant sequences of operations, *e.g.*

$$DD \ ; \ \textit{timer-start} \ ; (\sqsubseteq \ ; \sqcap ; \ := \ ; \sqcup)^* ; \ \textit{min} \ ; \ \textit{timer-stop}$$

- run the sequence on each library with random inputs

Ongoing Work II (Hang YU)

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

Parallelization, floating-point computations then reconstruction

- minimization by raytracing
  independant determination of each constraint
- the Solver of Parametric Linear Problems
  (C, C++, GLPK, FLINT, EIGEN, COQ, OCAML)
- new algorithm for inclusion

Will someday Polyhedra be usable?

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

May be with

- a polyhedra-aware static analyzer
- polyhedra used in a second phase where intervals failed
- dynamic packing of variables, removal of useless variable
- algorithms in constraint-only can benefit libraries in DD: the costly Chernikova's conversion can be delayed

Will someday Polyhedra be usable?

The field of polyhedra still makes progress

Nice result by [Singh et al., 2017] based on [Halbwachs et al., 2006] to cope with the hypercube phenomenon:

$$\begin{aligned} \mathcal{H} &= \mathrm{DD}(\textstyle\bigwedge_{i=1}^{n} -1 \leq x_i \leq 1) \\ &= (2 \times n \text{ constraints}, 2^n \text{ generators}) \end{aligned}$$

- The Fast Polyhedra Abstract Domain automatically splits polyhedra into a cartesian product during the analysis.
- Fast polyhedra almost behave like intervals when variables are not related.

Cartesian product of polyhedra [Halbwachs+1996, Singh+2015]

$$\mathcal{H} = P_1 \times \ldots \times P_n \text{ where } P_i = (\ \{-1 \leq x_i \leq 1\}\ ,\ \{-1, 1\}\ )$$

References of inspiring work I

Benoy, F., King, A., and Mesnard, F. (2005).
Computing convex hulls with a linear solver.
Theory and Practice of Logic Programming (TPLP), 5(1-2):259–271.

Besson, F., Jensen, T. P., Pichardie, D., and Turpin, T. (2010).
Certified result checking for polyhedral analysis of bytecode programs.
In Trustworthy Global Computing (TGC), volume 6084, pages 253–267. Springer.

Cousot, P. and Halbwachs, N. (1978).
Automatic discovery of linear restraints among variables of a program.
In ACM Principles of Programming Languages (POPL), pages 84–97. ACM Press.

Feautrier, P. (1988).
Parametric integer programming.
RAIRO Recherche opérationnelle, 22(3):243–268.

Feautrier, P. and Lengauer, C. (2011).
Polyhedron model.
In Encyclopedia of Parallel Computing, volume 1, pages 1581–1592. Springer.

Halbwachs, N., Merchat, D., and Gonnord, L. (2006).
Some ways to reduce the space dimension in polyhedra computations.
Formal Methods in System Design, 29(1):79–95.

Howe, J. M. and King, A. (2012).
Polyhedral analysis using parametric objectives.
In Static Analysis Symposium (SAS), volume 7460 of LNCS, pages 41–57.

References of inspiring work II

Jeannet, B. and Miné, A. (2009).
APRON: A library of numerical abstract domains for static analysis.
In Computer Aided Verification (CAV), volume 5643 of LNCS, pages 661–667.

Jones, C., N., Kerrigan, E. C., and Maciejowski, J. M. (2008).
On polyhedral projections and parametric programming.
Jounral of Optimization Theory and Applications, 138(2):207–220.

Miné, A. (2006).
Symbolic methods to enhance the precision of numerical abstract domains.
In Verification, Model Checking, and Abstract Interpretation (VMCAI), volume 3855 of LNCS, pages 348–363. Springer.

Singh, G., Püschel, M., and Vechev, M. (2017).
Fast polyhedra abstract domain.
In ACM Principles of Programming Languages (POPL), pages 46–59. ACM Press.

Wadler, P. (1989).
Theorems for free!
In Functional Programming Languages and Computer Architecture (FPLCA), pages 347–359. ACM Press.

Publications related to the VPL

Verimag Verified Polyhedra Library

Introduction
The VPL
Convex Polyhedra

Certification
Why ? How ?
VPL correctness
Farkas Lemma

The certified checker
Certification in COQ
Usage in VPL

Computation
Representation of Polyhedra
New algorithms
Experiments

Ongoing Work

Conclusion
Related work

on the algorithmic side of the VPL

SAS'2013 *Efficient generation of correctness certificates for the abstract domain of polyhedra*

VMCAI'2016 *Polyhedral Approximation of Multivariate Polynomials using Handelman's Theorem*

VMCAI'2017 *Efficient Elimination of Redundancies in Polyhedra using Raytracing*

SAS'2017 *Scalable Minimizing-Operators on Polyhedra via Parametric Linear Programming*

on the COQ side of the VPL

ITP'2015 *Refinement to Certify Abstract Interpretations, Illustrated on Linearization for Polyhedra*

VSTTE'2014 *A Certifying Frontend for (Sub)polyhedral Abstract Domains*