Possibly Not Closed Convex Polyhedra and the Parma Polyhedra Library

Roberto BAGNARA, Elisa RICCI, Enea ZAFFANELLA
University of Parma, Italy
Patricia M. HILL
University of Leeds, United Kingdom

http://www.cs.unipr.it/ppl/

Plan of the Talk

① Convex Polyhedra: the Double Description Method

② Not Necessarily Closed Polyhedra

  ① High-Level Interface

  ② Low-Level Representation

  ③ Minimization Issues

③ The Parma Polyhedra Library

Convex Polyhedra: What and Why

What?

- ➜ regions of $\mathbb{R}^n$ bounded by a finite set of hyperplanes.

Why? Solving Classical Data-Flow Analysis Problems!

- ➜ array bound checking and compile-time overflow detection;
- ➜ loop invariant computations and loop induction variables.

Why? Verification of Concurrent and Reactive Systems!

- ➜ synchronous languages;
- ➜ linear hybrid automata (roughly, FSMs with time requirements);
- ➜ systems based on temporal specifications.

And Again: Many Other Applications...

- ➜ inferring argument size relationships in logic programs;
- ➜ termination inference for Prolog programs;
- ➜ string cleanness for C programs.

The Double Description Method by Motzkin et al.

Constraint Representation: $\mathrm{con}(\mathcal{C})$

- If $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{a} \neq \mathbf{0}$, and $b \in \mathbb{R}$, the linear inequality constraint $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \geq b$ defines a closed affine half-space.
- All *closed polyhedra* can be expressed as the conjunction of a finite number of such constraints.

Generator Representation: $\mathrm{gen}(\mathcal{G})$, where $\mathcal{G} = (R, P)$

- If $\mathcal{P} \subseteq \mathbb{R}^n$ and $\mathcal{P} \neq \varnothing$, a vector $\boldsymbol{r} \in \mathbb{R}^n$ such that $\boldsymbol{r} \neq \mathbf{0}$ is a *ray of* $\mathcal{P}$ iff for each point $\boldsymbol{p} \in \mathcal{P}$ and each $\lambda \in \mathbb{R}_+$, we have $\boldsymbol{p} + \lambda \boldsymbol{r} \in \mathcal{P}$.
- All *closed polyhedra* can be expressed as

$$\left\{\, R\boldsymbol{\rho} + P\boldsymbol{\pi} \in \mathbb{R}^n \;\middle|\; \boldsymbol{\rho} \in \mathbb{R}^r_+, \boldsymbol{\pi} \in \mathbb{R}^p_+, \textstyle\sum_{i=1}^p \pi_i = 1 \,\right\}.$$

Advantages of the Dual Description Method

Some Operations Are More Efficiently Performed on Constraints

- ➜ Intersection is implemented as the union of constraint systems.
- ➜ Adding constraints (of course).

Some Operations Are More Efficiently Performed on Generators

- ➜ Convex polyhedral hull (poly-hull): union of generator systems.
- ➜ Adding generators (of course).
- ➜ Projection (i.e., removing dimensions).
- ➜ Finiteness (boundedness) check.
- ➜ Time-elapse.

Some Operations Are More Efficiently Performed with Both

- ➜ Minimization of the descriptions.
- ➜ Inclusion and equality tests.
- ➜ Widening.

Not Necessarily Closed Polyhedra

On the Constraints Side: Strict Inequalities

- If $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{a} \neq \mathbf{0}$, and $b \in \mathbb{R}$, the linear strict inequality constraint $\langle \boldsymbol{a}, \boldsymbol{x} \rangle > b$ defines an open affine half-space.
- Mixed constraint systems $\Longleftrightarrow$ NNC polyhedra.

Not Necessarily Closed Polyhedra

On the Constraints Side: Strict Inequalities

- ➜ If $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{a} \neq \mathbf{0}$, and $b \in \mathbb{R}$, the linear strict inequality constraint $\langle \boldsymbol{a}, \boldsymbol{x} \rangle > b$ defines an open affine half-space.
- ➜ Mixed constraint systems $\Longleftrightarrow$ NNC polyhedra.

On the Generators Side?

Not Necessarily Closed Polyhedra

On the Constraints Side: Strict Inequalities

- If $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{a} \neq \mathbf{0}$, and $b \in \mathbb{R}$, the linear strict inequality constraint $\langle \boldsymbol{a}, \boldsymbol{x} \rangle > b$ defines an open affine half-space.
- Mixed constraint systems $\Longleftrightarrow$ NNC polyhedra.

On the Generators Side?

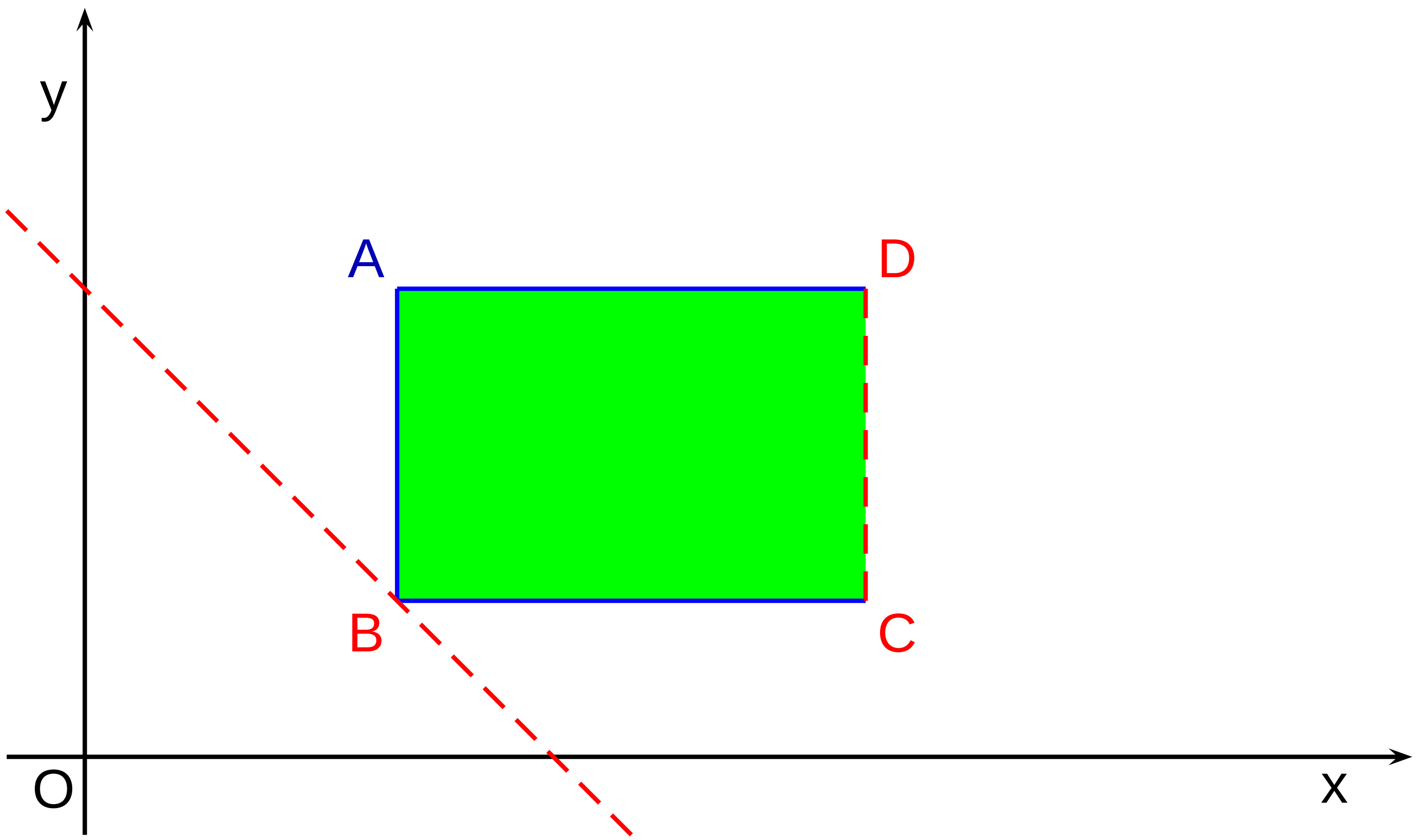
- A vector $\boldsymbol{c} \in \mathbb{R}^n$ is a closure point of the NNC polyhedron $\mathcal{P} \subseteq \mathbb{R}^n$ if and only if $\mathcal{P} \neq \varnothing$ and for every point $\boldsymbol{p} \in \mathcal{P}$ and $\lambda \in \mathbb{R}$ such that $0 < \lambda < 1$, it holds $\lambda \boldsymbol{p} + (1 - \lambda)\boldsymbol{c} \in \mathcal{P}$.
- All *NNC polyhedra* can be expressed as

$$\left\{ R\boldsymbol{\rho} + P\boldsymbol{\pi} + C\boldsymbol{\gamma} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \boldsymbol{\rho} \in \mathbb{R}^r_+, \boldsymbol{\pi} \in \mathbb{R}^p_+, \boldsymbol{\gamma} \in \mathbb{R}^c_+, \\ \boldsymbol{\pi} \neq \mathbf{0}, \sum_{i=1}^p \pi_i + \sum_{i=1}^c \gamma_i = 1 \end{array} \right\}.$$

- Extended generator systems $\Longleftrightarrow$ NNC polyhedra.

Example Using Constraints

$\mathcal{P} = \text{con}\big(\{2 \leq x, x < 5, 1 \leq y \leq 3, x + y > 3\}\big).$

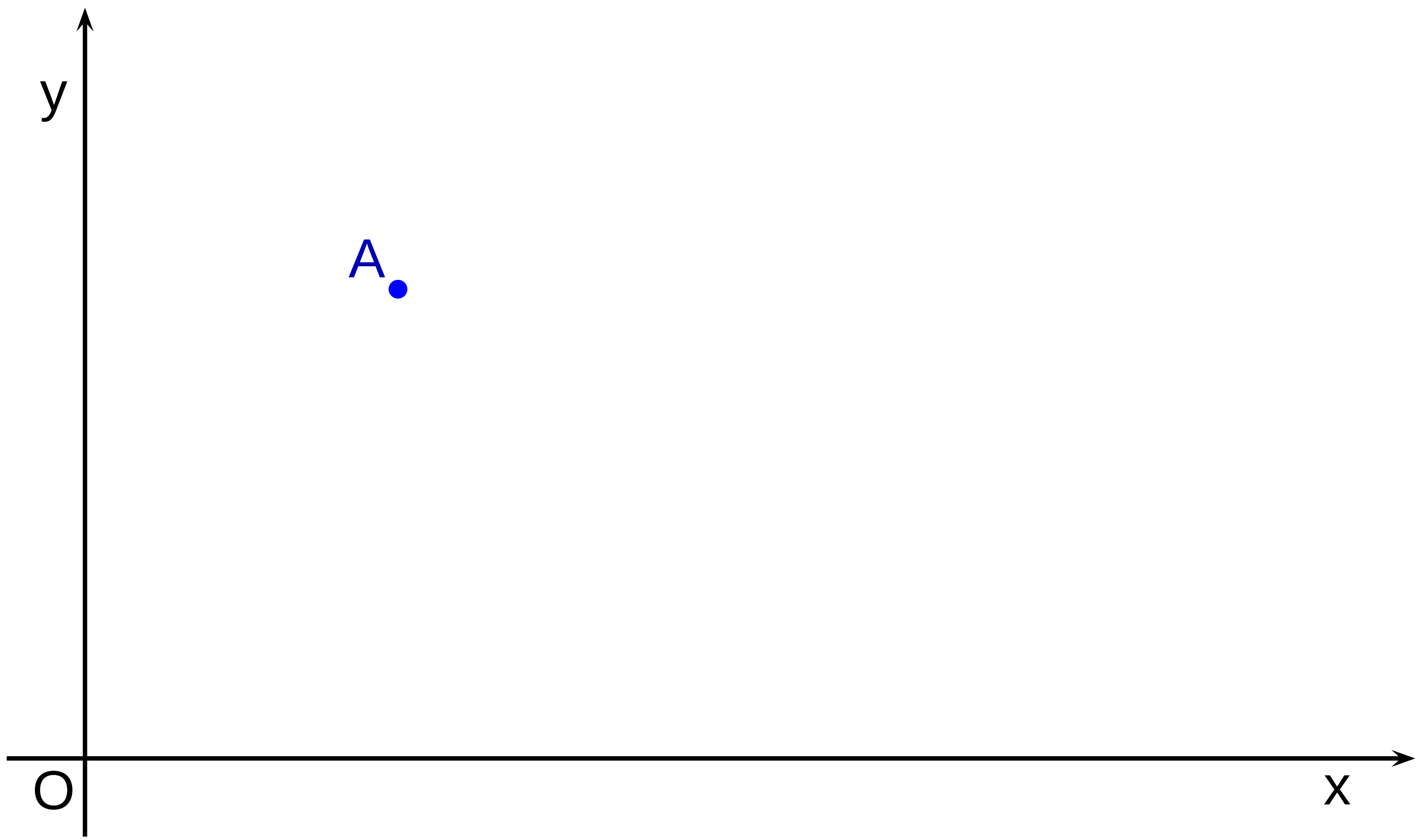


Same Example Using Generators (i)

$$\mathcal{P} = \mathrm{gen}\big((R, P, C)\big) = \mathrm{gen}\big((\varnothing, \varnothing, \varnothing)\big).$$

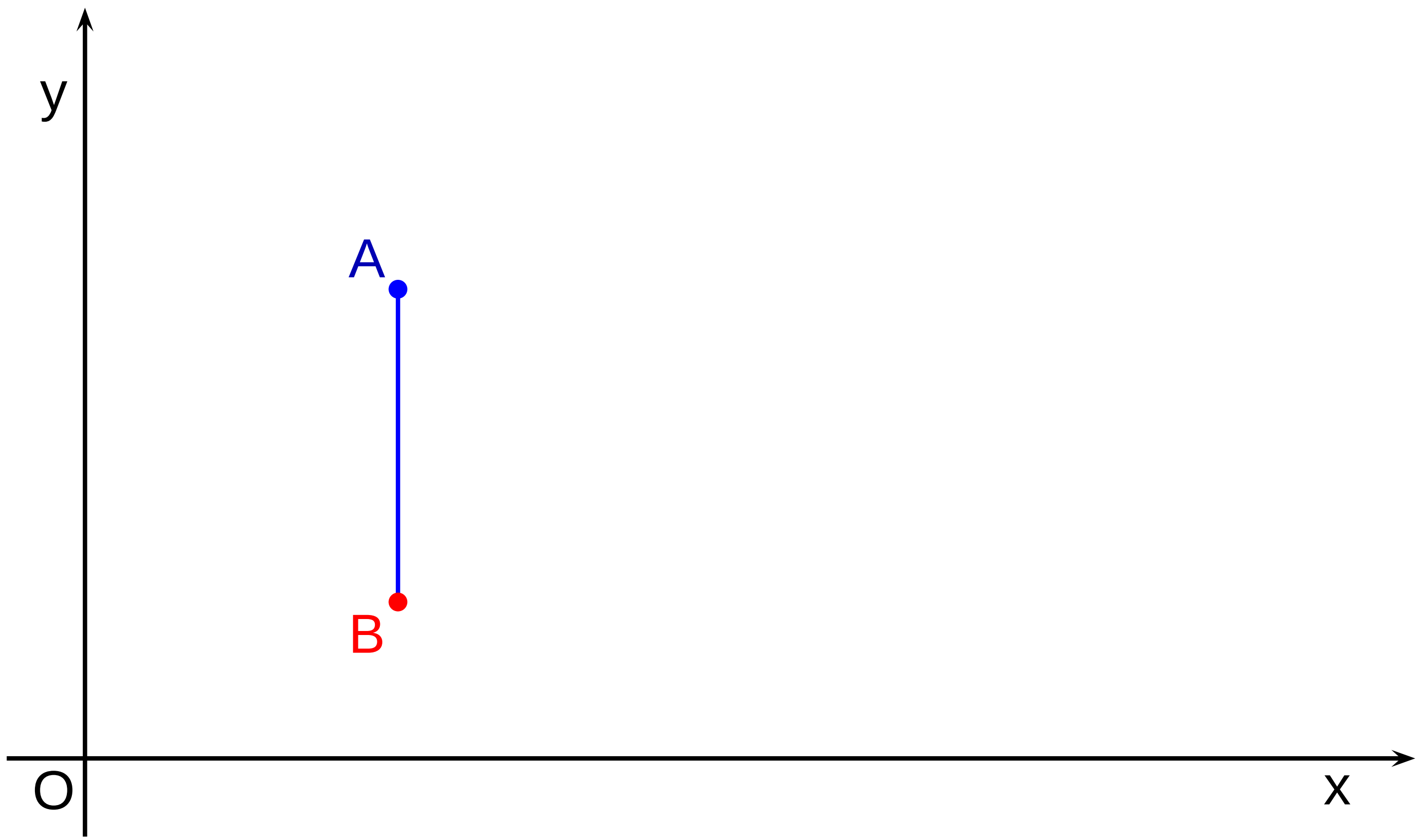
Same Example Using Generators (ii)

$$\mathcal{P} = \text{gen}\big((R, P, C)\big) = \text{gen}\Big(\big(\varnothing, \{A\}, \varnothing\big)\Big).$$



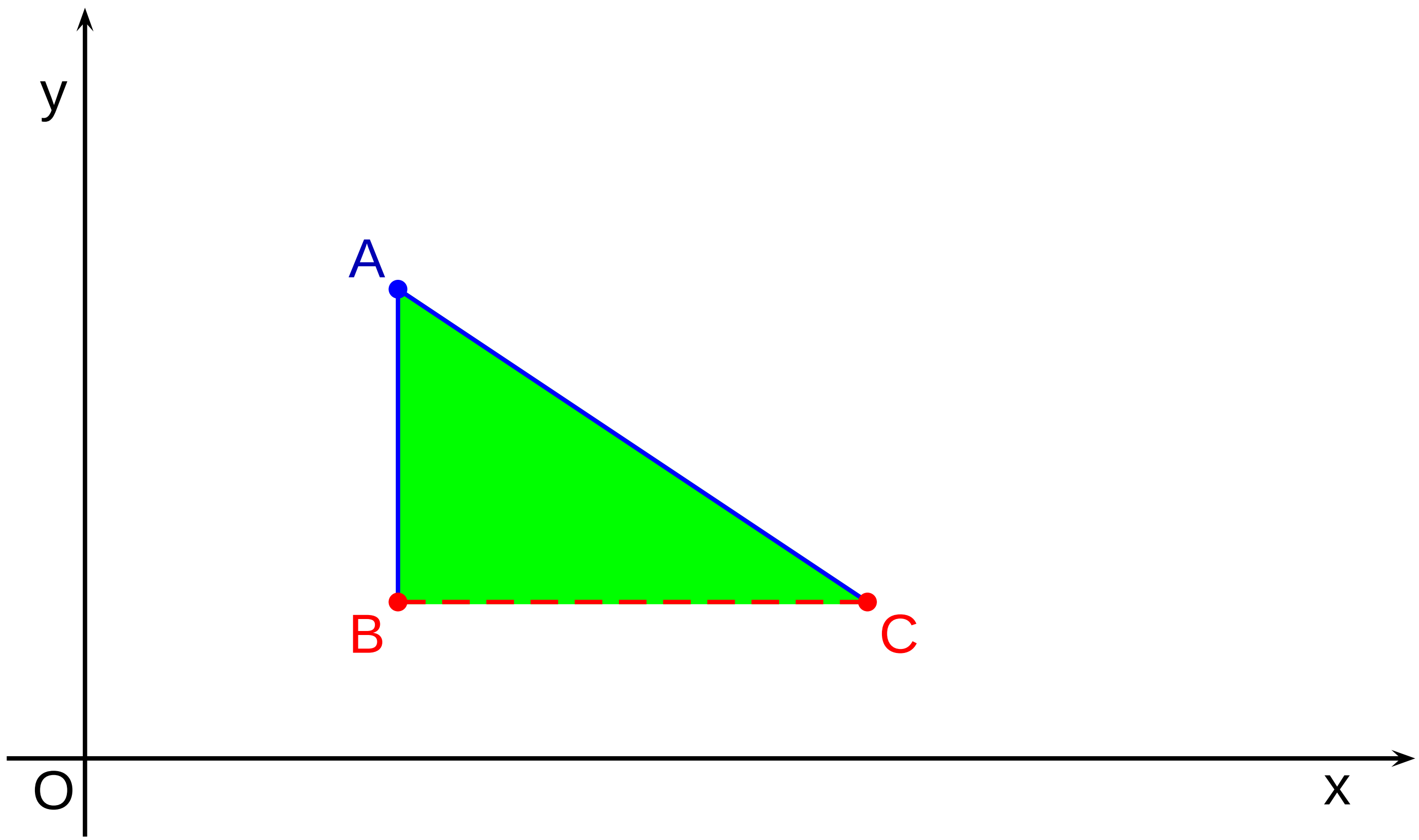
Same Example Using Generators (iii)

$$\mathcal{P} = \mathrm{gen}\big((R, P, C)\big) = \mathrm{gen}\Big(\big(\varnothing, \{A\}, \{B\}\big)\Big).$$



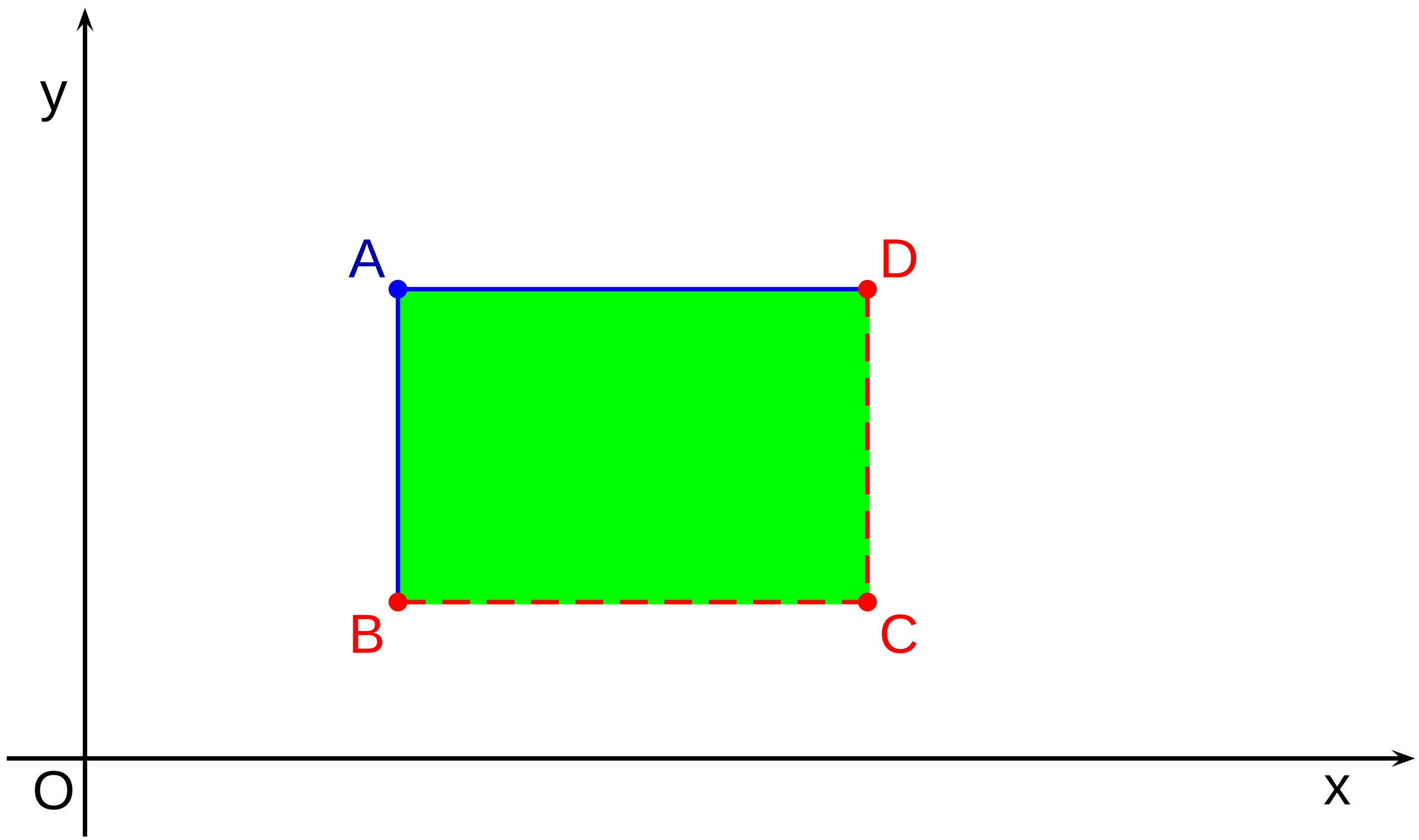
Same Example Using Generators (iv)

$$\mathcal{P} = \text{gen}\big((R, P, C)\big) = \text{gen}\Big(\big(\varnothing, \{A\}, \{B, C\}\big)\Big).$$



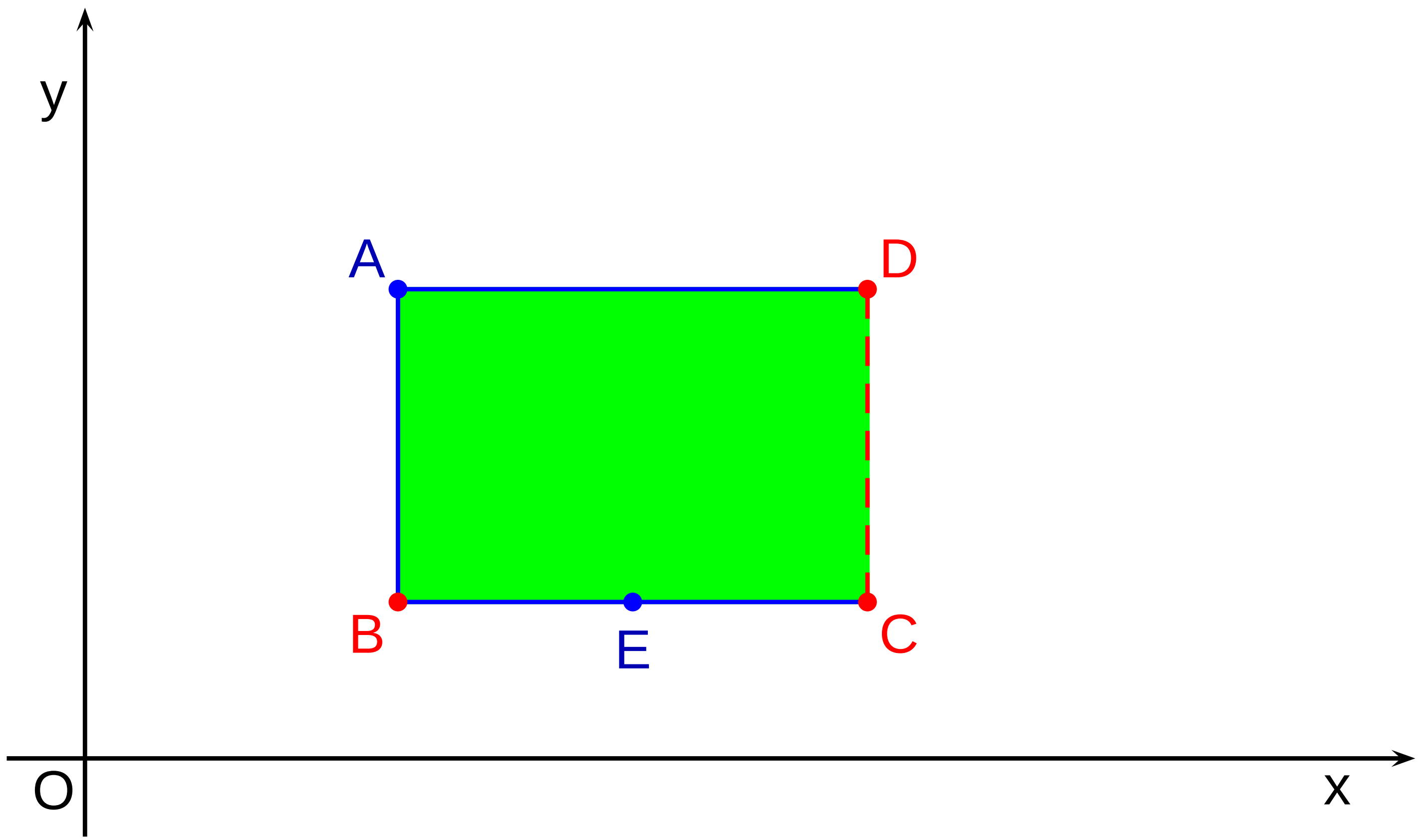
Same Example Using Generators (v)

$$\mathcal{P} = \mathrm{gen}\big((R, P, C)\big) = \mathrm{gen}\Big(\big(\varnothing, \{A\}, \{B, C, D\}\big)\Big).$$



Same Example Using Generators (vi)

$$\mathcal{P} = \mathrm{gen}\big((R, P, C)\big) = \mathrm{gen}\Big(\big(\varnothing, \{A, E\}, \{B, C, D\}\big)\Big).$$



Encoding NNC Polyhedra as C Polyhedra

- Let $\mathbb{P}_n$ and $\mathbb{CP}_n$ be the sets of all NNC and closed polyhedra, respectively: each $\mathcal{P} \in \mathbb{P}_n$ can be embedded into $\mathcal{R} \in \mathbb{CP}_{n+1}$.
- If $\mathcal{P} \in \mathbb{P}_n$ and $\mathcal{P} = \mathrm{con}(\mathcal{C})$, where

$$\mathcal{C} = \bigl\{ \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \bowtie_i b_i \bigm| i \in \{1, \ldots, m\}, \boldsymbol{a}_i \in \mathbb{R}^n, \bowtie_i \in \{\geq, >\}, b_i \in \mathbb{R} \bigr\},$$

then $\mathcal{R} \in \mathbb{CP}_{n+1}$ is defined by $\mathcal{R} = \mathrm{con}\bigl(\mathrm{con\_repr}(\mathcal{C})\bigr)$, where

$$\begin{aligned} \mathrm{con\_repr}(\mathcal{C}) \stackrel{\mathrm{def}}{=} \{0 \leq \epsilon \leq 1\} & \\ \cup \bigl\{ \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - 1 \cdot \epsilon \geq b_i \bigm| i \in \{1, \ldots, m\}, \bowtie_i \in \{>\} \bigr\} & \\ \cup \bigl\{ \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle + 0 \cdot \epsilon \geq b_i \bigm| i \in \{1, \ldots, m\}, \bowtie_i \in \{\geq\} \bigr\}. & \end{aligned}$$

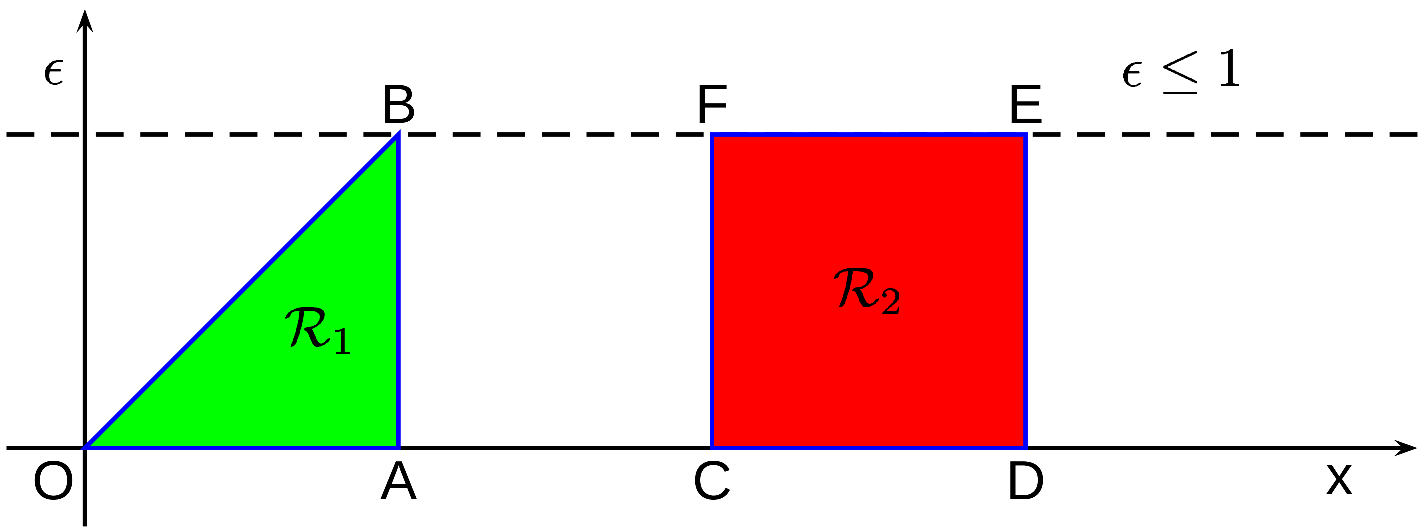
- The encoded NNC polyhedron:

$$\mathcal{P} = [\![\mathcal{R}]\!] \stackrel{\mathrm{def}}{=} \bigl\{ \boldsymbol{v} \in \mathbb{R}^n \bigm| \exists e > 0 \,.\, (\boldsymbol{v}^{\mathrm{T}}, e)^{\mathrm{T}} \in \mathcal{R} \bigr\}.$$

EXAMPLE: ENCODING $\mathbb{P}_1$ INTO $\mathbb{CP}_2$

$\mathcal{R}_1$ encodes $\mathcal{P}_1 = \mathrm{con}\big(\{0 < x \leq 1\}\big)$,

$\mathcal{R}_2$ encodes $\mathcal{P}_2 = \mathrm{con}\big(\{2 \leq x \leq 3\}\big)$.



Same Encoding Using Generators

- From the New Polka manual (s is the ϵ coefficient):

  Don't ask me the intuitive meaning of $s \neq 0$ in rays and vertices !

- From the Polka manual:

  While strict inequations handling is transparent for constraints [...] the extra dimension added to the variables space is apparent when it comes to generators [...]

- If $\mathcal{P} \in \mathbb{P}_n$ and $\mathcal{P} = \mathrm{gen}(\mathcal{G})$, where $\mathcal{G} = (R, P, C)$, then $\mathcal{R} \in \mathbb{CP}_{n+1}$ is defined by $\mathcal{R} = \mathrm{gen}\bigl(\mathrm{gen\_repr}(\mathcal{G})\bigr) = \mathrm{gen}\bigl((R', P')\bigr)$, where

$$R' = \bigl\{\, (\boldsymbol{r}^{\mathrm{T}}, 0)^{\mathrm{T}} \bigm| \boldsymbol{r} \in R \,\bigr\},$$

$$P' = \bigl\{\, (\boldsymbol{p}^{\mathrm{T}}, 1)^{\mathrm{T}}, (\boldsymbol{p}^{\mathrm{T}}, 0)^{\mathrm{T}} \bigm| \boldsymbol{p} \in P \,\bigr\} \cup \bigl\{\, (\boldsymbol{c}^{\mathrm{T}}, 0)^{\mathrm{T}} \bigm| \boldsymbol{c} \in C \,\bigr\}.$$

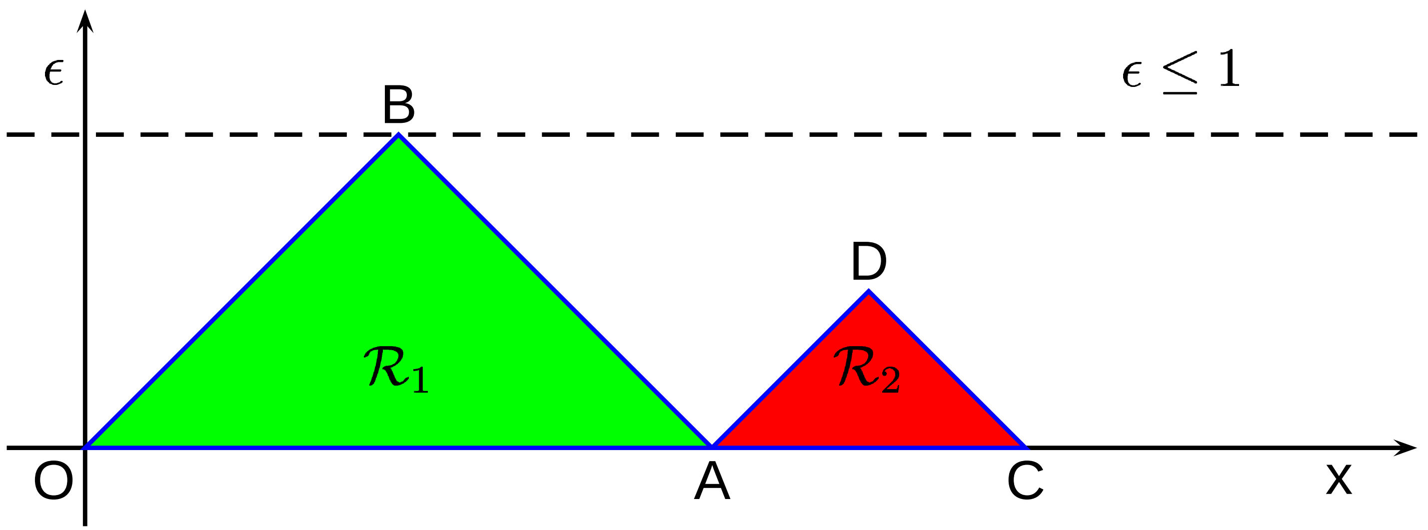
Minimization of NNC Polyhedra

- The problem: in no way does minimization of the representation in $\mathbb{CP}_{n+1}$ imply minimization of the NNC polyhedron in $\mathbb{P}_n$.
- There are examples where a “minimized” representation has more than half of the constraints that are redundant.
- This causes both efficiency and usability problems:
  - the client application must distinguish between the real constraints/generators and the surrounding noise.

Example (i)

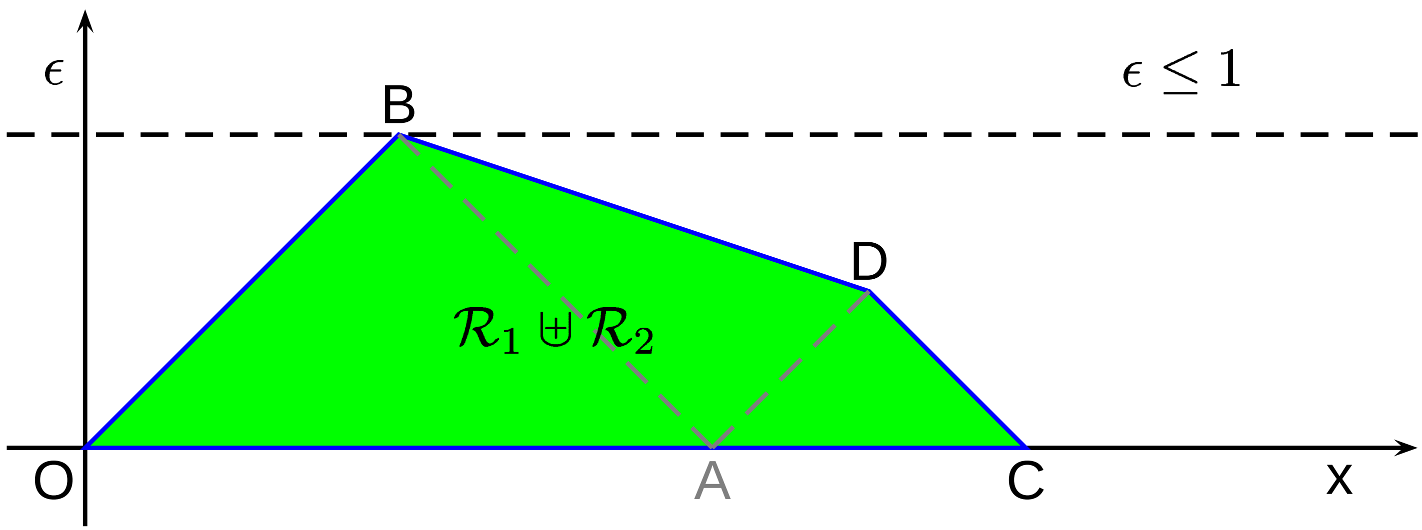
$\mathcal{R}_1$ encodes $\mathcal{P}_1 = \mathrm{con}\big(\{0 < x < 2\}\big)$,

$\mathcal{R}_2$ encodes $\mathcal{P}_2 = \mathrm{con}\big(\{2 < x < 3\}\big)$.



Example (II)

$\mathcal{R}_1 \uplus \mathcal{R}_2$ encodes the poly-hull $\mathcal{P}_1 \uplus \mathcal{P}_2 = \mathrm{con}\big(\{0 < x < 3\}\big)$, but segment $[B, D]$ also encodes the redundant constraint $x < 4$.

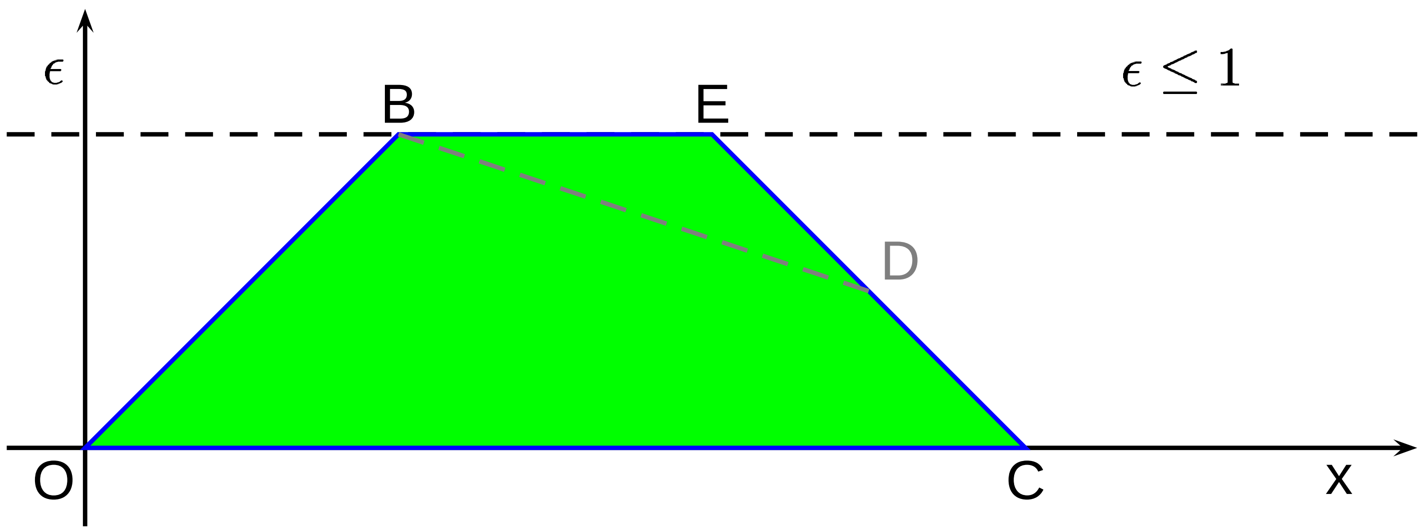


STRONG MINIMAL FORMS FOR $\mathbb{CP}_{n+1}$ ENCODINGS

- ➜ The goal: provide a constraint/generator description such that none of its proper subsets represents the same NNC polyhedron.
- ➜ The solution: remove the ϵ-redundant constraints/generators.
- ➜ The implementation: rather efficient ϵ-redundancy test based on saturation conditions (reusing partial results of standard minimization algorithm).

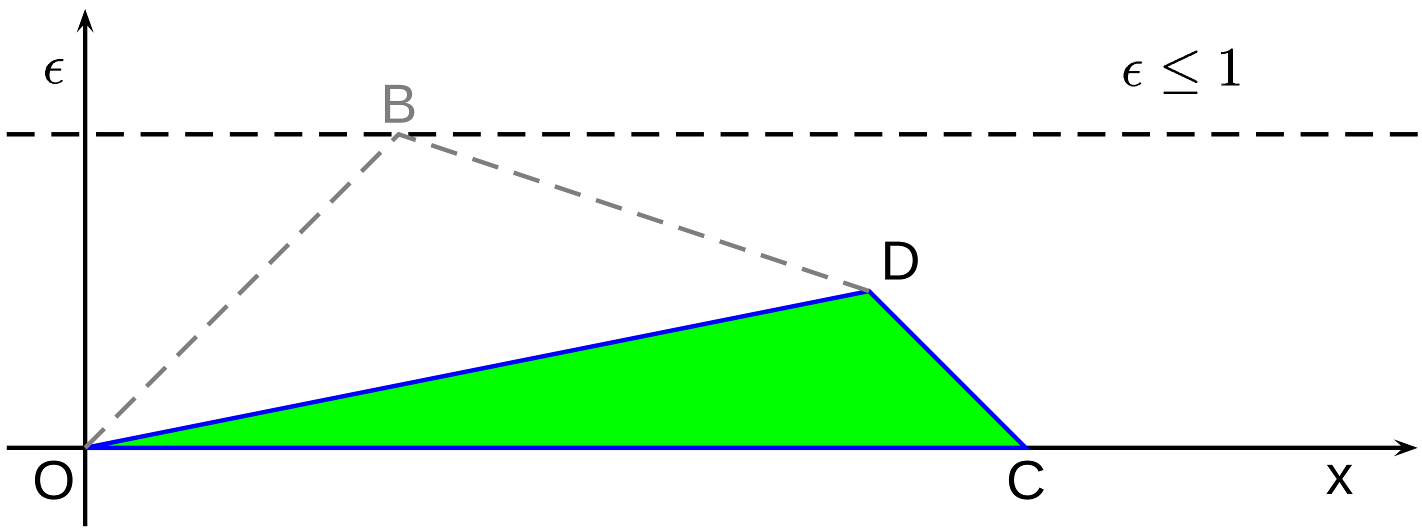
Strong Minimal Forms for $\mathbb{CP}_{n+1}$ Encodings

- ➜ The goal: provide a constraint/generator description such that none of its proper subsets represents the same NNC polyhedron.
- ➜ The solution: remove the ϵ-redundant constraints/generators.
- ➜ The implementation: rather efficient ϵ-redundancy test based on saturation conditions (reusing partial results of standard minimization algorithm).



Strong Minimal Forms for $\mathbb{CP}_{n+1}$ Encodings

- The goal: provide a constraint/generator description such that none of its proper subsets represents the same NNC polyhedron.
- The solution: remove the ϵ-redundant constraints/generators.
- The implementation: rather efficient ϵ-redundancy test based on saturation conditions (reusing partial results of standard minimization algorithm).



THE PARMA POLYHEDRA LIBRARY

➜ A collaborative project started in January 2001 at the Department of Mathematics of the University of Parma.

➜ Aims at the development of a truly professional and free library for the handling of (NNC) rational convex polyhedra, targeted at abstract interpretation and computer-aided verification.

The Parma Polyhedra Library

- ➜ A collaborative project started in January 2001 at the Department of Mathematics of the University of Parma.
- ➜ Aims at the development of a truly professional and free library for the handling of (NNC) rational convex polyhedra, targeted at abstract interpretation and computer-aided verification.

Why Yet Another Library? Some Limitations of Existing Ones:

- ➜ data-structures employed cannot grow/shrink dynamically;
- ➜ possibility of overflow, underflow and rounding errors;
- ➜ unsuitable mechanisms for error detection, handling and recovery;
  - ➜ (cannot reliably resume computation with an alternative method, e.g., by reverting to an interval-based approximation).
- ➜ Several existing libraries are free, but they do not provide documentation for the interfaces and code that is adequate for an outsider to make improvements with any real confidence.

PPL Features

Portability Across Different Computing Platforms

- ➜ written in standard C++;
- ➜ but the the client application needs not be written in C++.

Absence of Arbitrary Limits

- ➜ arbitrary precision integer arithmetic for coefficients and coordinates;
- ➜ all data structures can expand automatically (in amortized constant time) to any dimension allowed by the available virtual memory.

Complete Information Hiding

- ➜ the internal representation of constraints, generators and systems thereof need not concern the client application;
- ➜ implementation devices such as the *positivity constraint* as well as all the matters regarding the ϵ-representation encoding of NNC polyhedra are invisible from outside.

PPL Features: Hiding Pays

Expressivity

- ‘`X + 2*Y + 5 >= 7*Z`’ and ‘`ray(3*X + Y)`’ is valid syntax both for the C++ and the Prolog interfaces;
- the planned Objective Caml and Mercury interfaces will be as friendly as these;
- even the C interface refers to high-level concepts and not to their possible implementation (vectors, matrices, etc.).

Failure Avoidance and Detection

- illegal objects cannot be created easily;
- the interface invariants are systematically checked.

Efficiency

- can systematically apply incremental and lazy computation techniques.

PPL Features: Support for Robustness

```
void complex_function(PH& ph1, const PH& ph2 ...) {
  try {
    start_timer(max_time_for_complex_function);
    complex_function_on_polyhedra(ph1, ph2 ...);
    stop_timer();
  }
  catch (Exception& e) { // Out of memory or timeout...
    BoundingBox bb1, bb2;
    ph1.shrink_bounding_box(bb1);
    ph2.shrink_bounding_box(bb2);
    complex_function_on_bounding_boxes(bb1, bb2 ...);
    ph1 = Polyhedron(bb1);
  }
}
```

Conclusion

- ➜ Convex polyhedra are the basis for several abstractions used in static analysis and computer-aided verification of complex and sometimes mission critical systems.
- ➜ For that purposes an implementation of convex polyhedra must be firmly based on a clear theoretical framework and written in accordance with sound software engineering principles.
- ➜ In this talk we have presented some of the most important ideas that are behind the Parma Polyhedra Library.
- ➜ The Parma Polyhedra Library is free software released under the GPL: code and documentation can be downloaded and its development can be followed at `http://www.cs.unipr.it/ppl/`.

The Inclusion Test for Closed Polyhedra

- Let $\mathcal{P}_1 = \mathrm{gen}(\mathcal{G}_1) \in \mathbb{CP}_n$ and $\mathcal{P}_2 = \mathrm{con}(\mathcal{C}_2) \in \mathbb{CP}_n$.
- $\mathcal{P}_1 \subseteq \mathcal{P}_2$ iff each generator in $\mathcal{G}_1$ satisfies each constraint in $\mathcal{C}_2$;
  - generator $\boldsymbol{g} \in \mathbb{R}^n$ satisfies constraint $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \bowtie b$ if and only if the scalar product $s \stackrel{\mathrm{def}}{=} \langle \boldsymbol{a}, \boldsymbol{g} \rangle$ satisfies the following condition:

	Constraint type	
Generator type	$=$	$\geq$
line	$s = 0$	$s = 0$
ray	$s = 0$	$s \geq 0$
point	$s = b$	$s \geq b$

The Inclusion Test for NNC Polyhedra

- Let $\mathcal{P}_1 = \mathrm{gen}(\mathcal{G}_1) \in \mathbb{P}_n$ and $\mathcal{P}_2 = \mathrm{con}(\mathcal{C}_2) \in \mathbb{P}_n$.
- $\mathcal{P}_1 \subseteq \mathcal{P}_2$ iff each generator in $\mathcal{G}_1$ satisfies each constraint in $\mathcal{C}_2$.
- Generator $\boldsymbol{g} \in \mathbb{R}^n$ satisfies constraint $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \bowtie b$ if and only if the scalar product $s \stackrel{\mathrm{def}}{=} \langle \boldsymbol{a}, \boldsymbol{g} \rangle$ satisfies the following condition:

	Constraint type		
Generator type	$=$	$\geq$	$>$
line	$s = 0$	$s = 0$	$s = 0$
ray	$s = 0$	$s \geq 0$	$s \geq 0$
point	$s = b$	$s \geq b$	$s > b$
closure point	$s = b$	$s \geq b$	$s \geq b$