Polyhedra to reason about software

Enea Zaffanella
Dept. of Mathematical, Physical and Computer Sciences
University of Parma (Italy)

Geometry and Computer Science, Pescara, Feb. 2017

Plan of the Talk

① Why formal methods in computer science
② Why polyhedral computations matter
③ The many facets of polyhedral computations
④ On the Detection of Exact Joins

Why Formal Methods in Computer Science

We Have a Problem

- Proper design, development and maintenance of computing systems (hardware and software) are expensive activities
- Size and complexity of computing systems are increasing
- Human resources are (more or less) stable
- Growing interest in any methodology that can assist the programmers

No Silver Bullet

- Domain specific, higher level languages
- Code inspections, reviews, audits
- Coding standards (language subsetting)
- Systematic testing
- Proofs of correctness
- ...
- No matter which methodologies are chosen, we need mechanical tools to help the programmer reasoning about programs

Example: Is `x/(x-y)` Well-Defined?

Many things may go wrong

- `x` and/or `y` may be uninitialized;
- `x` and `y` may be equal: division by 0;
- `x-y` may overflow;
- `x/x-y` may overflow;
- for floating point datatypes:
  - `x-y` may underflow;
  - `x/(x-y)` may underflow.

Example: Validation of Array References

Are these array accesses safe?

```
procedure shellsort(n : integer, array [0..n-1] of integer)
begin
  var h, i, j, B : integer;
  h := 1;
  while (h*3 + 1) < n do h := 3*h + 1;
  while h > 0 do
    i := h-1;
    while i < n do
      B := a[i]; j := i;
      while (j >= h) and (a[j-h] > B) do
        a[j] := a[j-h]; j := j-h;
      a[j] := B;
      i := i+1;
    h := h div 3;
```

Example: String Cleanness in C/C++

Taken from Web2c: an implementation of TeX and friends that translates the original WEB sources into C. See, `http://www.tug.org/web2c/`.

```
#define BUFSIZ 1024
char buf[BUFSIZ];

char* insert_long(char *cp) {
  char temp[BUFSIZ];
  int i;
  assert(cp >= buf[0] && cp < buf[BUFSIZ]);
  for (i = 0; &buf[i] < cp; ++i)
    temp[i] = buf[i];
  strcpy (&temp[i], "(long)");
  strcpy (&temp[i + 6], cp);
  strcpy (buf, temp);
  return cp + 6;
}
```

Formal Program Verification Methods

- To mechanically prove that all possible program executions are correct in all specified execution environments...
- ...for some definition of correct:
  - absence of certain kinds of run-time errors;
  - adherence to a (partial) specification...

Several methods

- program typing;
- deductive methods;
- static analysis;
- model checking.

Abstract Interpretation

Because of the undecidability of program verification

- ➜ all methods are partial or incomplete
- ➜ all resort to some form of approximation
- ➜ Abstract Interpretation is a framework to reason about sound approximation

P. Cousot, R. Cousot
Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints, *POPL 1977*

P. Cousot, R. Cousot
Abstract Interpretation Frameworks, *JLC 1992*

Abstract Interpretation

Assign a concrete meaning to a computing system

- $D = \langle C, \sqsubseteq, \bot, \top, \sqcap, \sqcup \rangle$, $\mathcal{F} \colon D \to D$
- $\bot \sqsubseteq \mathcal{F}(\bot) \sqsubseteq \ldots \sqsubseteq \mathcal{F}^i(\bot) \sqsubseteq \ldots$
- $M = \operatorname{lfp} \mathcal{F} = \bigsqcup \mathcal{F}^i(\bot)$

Assign an abstract meaning to the computing system

- $D^\sharp = \langle C^\sharp, \sqsubseteq^\sharp, \bot^\sharp, \top^\sharp, \sqcap^\sharp, \sqcup^\sharp \rangle$, $\mathcal{F}^\sharp \colon D^\sharp \to D^\sharp$
- $\bot^\sharp \sqsubseteq^\sharp \mathcal{F}^\sharp(\bot^\sharp) \sqsubseteq^\sharp \ldots \sqsubseteq^\sharp (\mathcal{F}^\sharp)^i(\bot^\sharp) \sqsubseteq^\sharp \ldots$
- $M^\sharp = \operatorname{lfp} \mathcal{F}^\sharp = \bigsqcup^\sharp (\mathcal{F}^\sharp)^i(\bot^\sharp)$

Soundness: the two meanings are related

- $\gamma \colon D^\sharp \to D$ (concretization function)
- $\bot \sqsubseteq \gamma(\bot^\sharp)$, $c \sqsubseteq \gamma(c^\sharp) \Rightarrow \mathcal{F}(c) \sqsubseteq \gamma(\mathcal{F}^\sharp(c^\sharp))$
- $M \sqsubseteq \gamma(M^\sharp)$

Why Polyhedral Computations Matter

A trivial example of static analysis

Example: The Concrete Meaning

```
x := 0; y := 0;

while x <= 100 do
```

$(x, y) \in M \in \wp(\mathbb{R}^2)$

```
  read(b);
  if b then x := x+2
  else x := x+1; y := y+1;
  endif
endwhile
```

Concrete domain:

$$\langle \wp(\mathbb{R}^2), \subseteq, \varnothing, \mathbb{R}^2, \cup, \cap \rangle.$$

Concrete meaning:

$$M \stackrel{\text{def}}{=} \operatorname{lfp} \mathcal{F} = \mathcal{F}^{\omega}(\varnothing).$$

Example: The Concrete Meaning

```
x := 0; y := 0;

while x <= 100 do
    ∅
  read(b);
  if b then x := x+2

  else x := x+1; y := y+1;

  endif

endwhile
```

Example: The Concrete Meaning

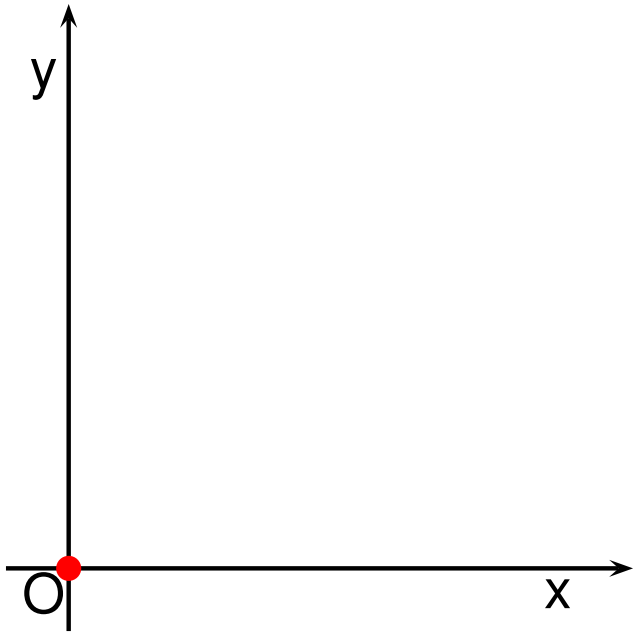
```
x := 0; y := 0;
```

$\{(0,0)\}$

```
while x <= 100 do
```

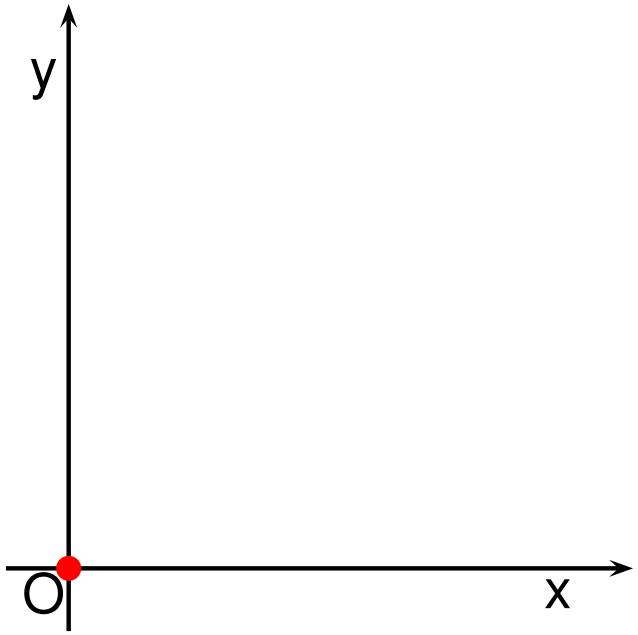
$\varnothing$

```
  read(b);
  if b then x := x+2

  else x := x+1; y := y+1;

  endif

endwhile
```



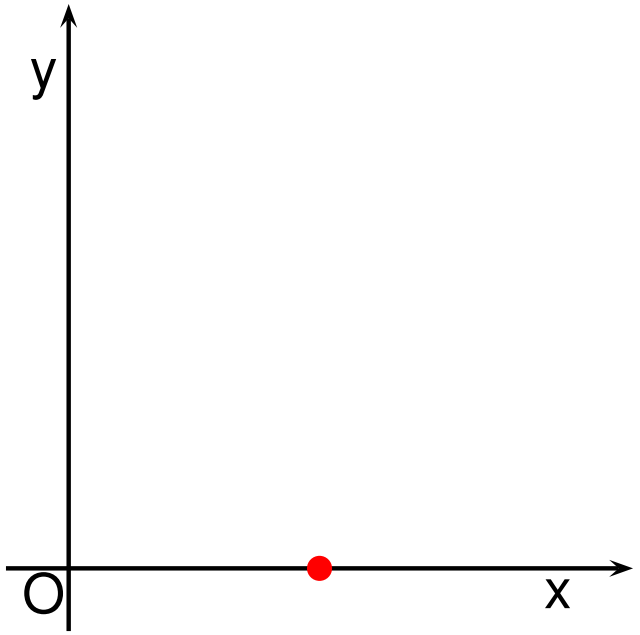
Example: The Concrete Meaning

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0)\}$
```
  read(b);
  if b then x := x+2

  else x := x+1; y := y+1;

  endif

endwhile
```



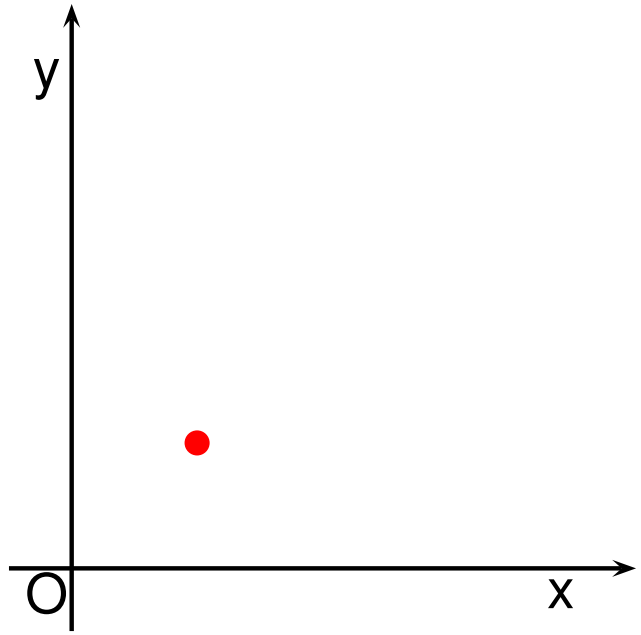
Example: The Concrete Meaning

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0)\}$
```
  read(b);
  if b then x := x+2
```
$\{(2,0)\}$
```
  else x := x+1; y := y+1;

  endif

endwhile
```



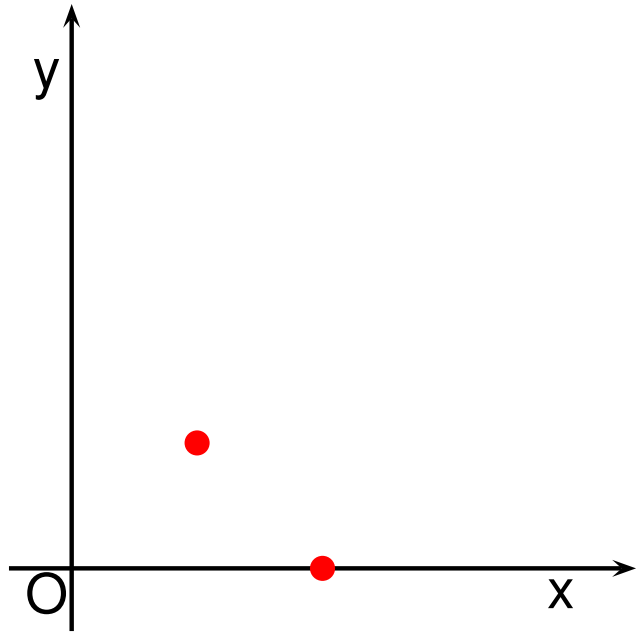
Example: The Concrete Meaning

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0)\}$
```
  read(b);
  if b then x := x+2
```
$\{(2,0)\}$
```
  else x := x+1; y := y+1;
```
$\{(1,1)\}$
```
  endif

endwhile
```



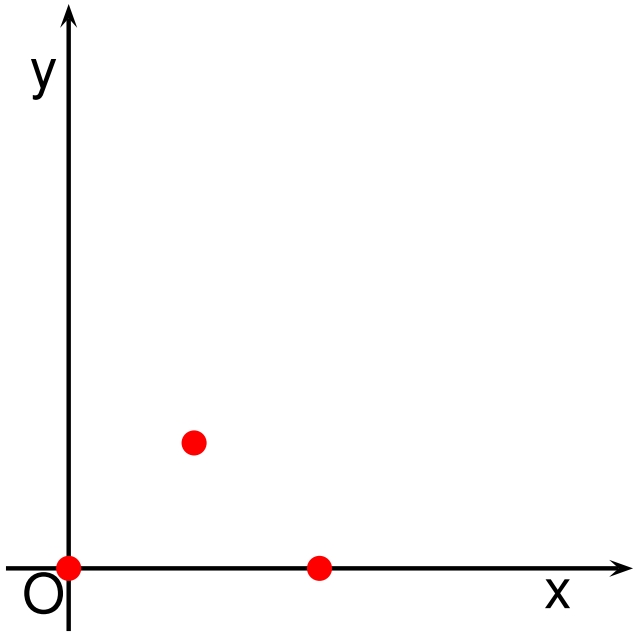
Example: The Concrete Meaning

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0)\}$
```
  read(b);
  if b then x := x+2
```
$\{(2,0)\}$
```
  else x := x+1; y := y+1;
```
$\{(1,1)\}$
```
  endif
```
$\{(1,1),(2,0)\}$
```
endwhile
```



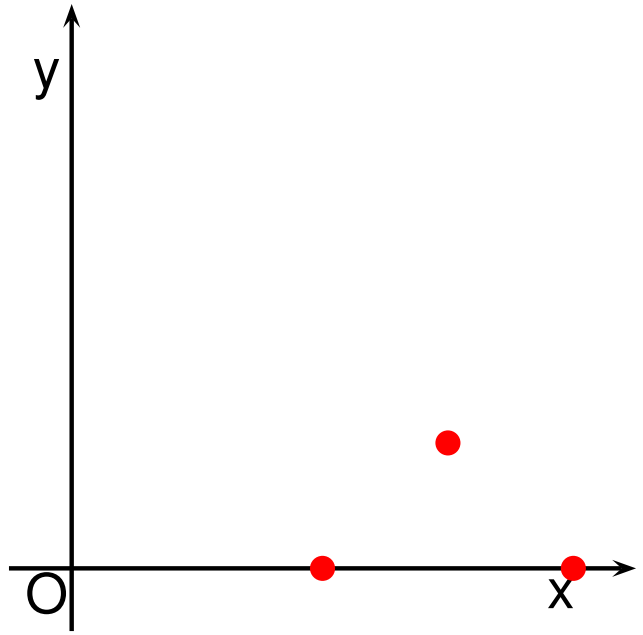
Example: The Concrete Meaning

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0),(1,1),(2,0)\}$
```
  read(b);
  if b then x := x+2
```
$\{(2,0)\}$
```
  else x := x+1; y := y+1;
```
$\{(1,1)\}$
```
  endif
```
$\{(1,1),(2,0)\}$
```
endwhile
```



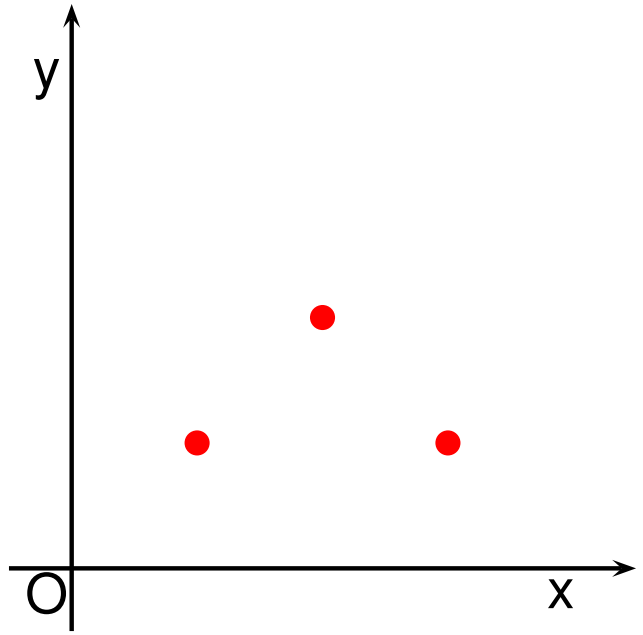
Example: The Concrete Meaning

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0),(1,1),(2,0)\}$
```
  read(b);
  if b then x := x+2
```
$\{(2,0),(3,1),(4,0)\}$
```
  else x := x+1; y := y+1;
```
$\{(1,1)\}$
```
  endif
```
$\{(1,1),(2,0)\}$
```
endwhile
```



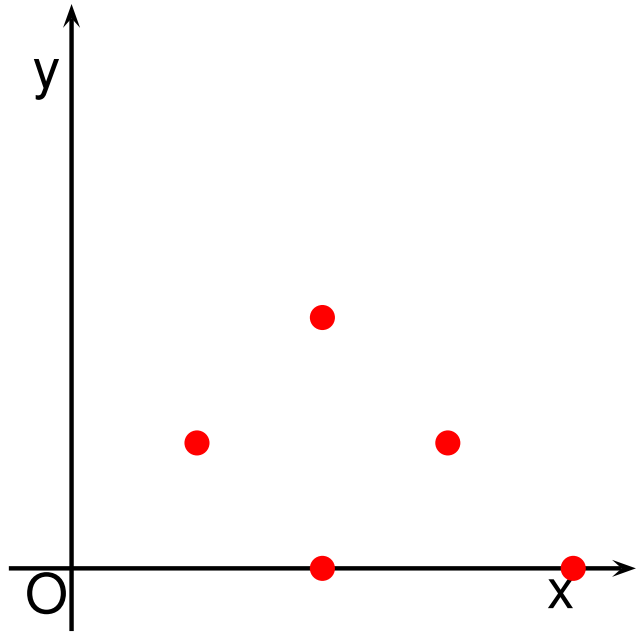
Example: The Concrete Meaning

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0),(1,1),(2,0)\}$
```
  read(b);
  if b then x := x+2
```
$\{(2,0),(3,1),(4,0)\}$
```
  else x := x+1; y := y+1;
```
$\{(1,1),(2,2),(3,1)\}$
```
  endif
```
$\{(1,1),(2,0)\}$
```
endwhile
```



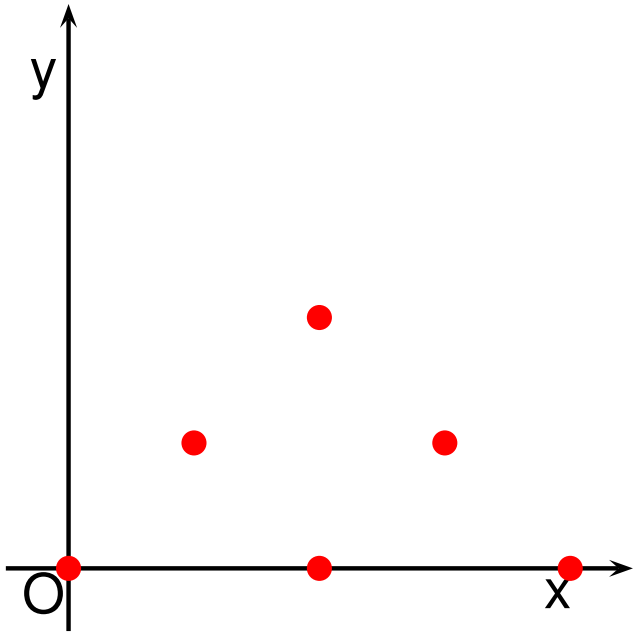
Example: The Concrete Meaning

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0),(1,1),(2,0)\}$
```
  read(b);
  if b then x := x+2
```
$\{(2,0),(3,1),(4,0)\}$
```
  else x := x+1; y := y+1;
```
$\{(1,1),(2,2),(3,1)\}$
```
  endif
```
$\{(1,1),(2,0),(2,2),(3,1),(4,0)\}$
```
endwhile
```



EXAMPLE: . . . AND SO ON . . .

```
x := 0; y := 0;
```
$\{(0,0)\}$
```
while x <= 100 do
```
$\{(0,0),(1,1),(2,0),(2,2),(3,1),(4,0)\}$
```
  read(b);
  if b then x := x+2
```
$\{(2,0),(3,1),(4,0)\}$
```
  else x := x+1; y := y+1;
```
$\{(1,1),(2,2),(3,1)\}$
```
  endif
```
$\{(1,1),(2,0),(2,2),(3,1),(4,0)\}$
```
endwhile
```



Example: The Abstract Meaning

```
x := 0; y := 0;

while x <= 100 do
```
$(x, y) \in \mathcal{Q} \in \mathbb{CP}_2$
```
  read(b);
  if b then x := x+2

  else x := x+1; y := y+1;

  endif

endwhile
```

Abstract domain:

$$\langle \mathbb{CP}_2, \subseteq, \varnothing, \mathbb{R}^2, \uplus, \cap \rangle.$$

Correctness:

$$X \subseteq \mathcal{P} \implies \mathcal{F}(X) \subseteq \mathcal{F}^\sharp(\mathcal{P}).$$

Abstract meaning:

$$\mathcal{Q} \in \mathrm{postfp}(\mathcal{F}^\sharp).$$

Example: The Abstract Meaning

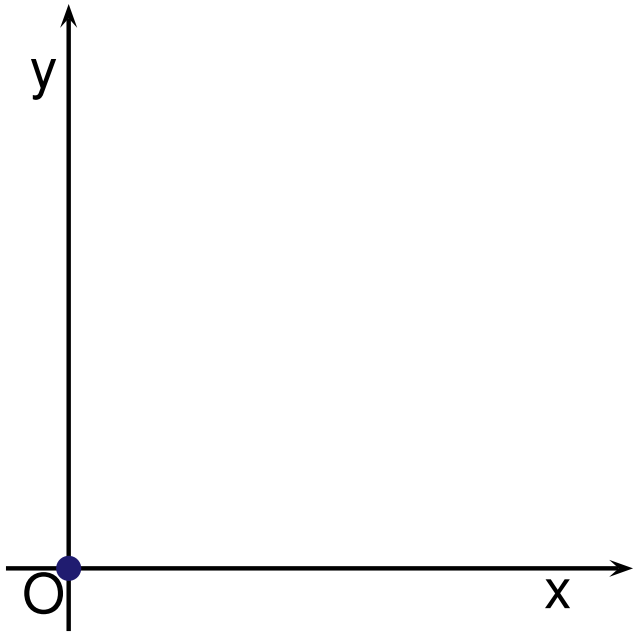
```
x := 0; y := 0;

while x <= 100 do
```

$\{1 = 0\}$

```
  read(b);
  if b then x := x+2

  else x := x+1; y := y+1;

  endif

endwhile
```

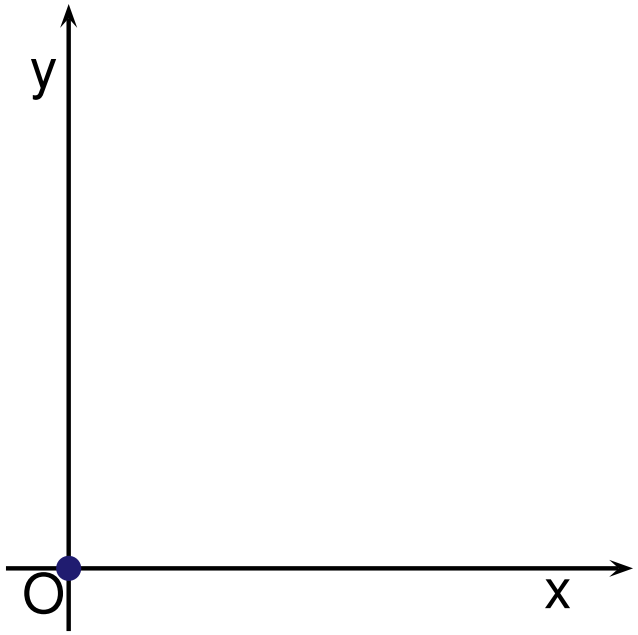
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{1 = 0\}$
```
  read(b);
  if b then x := x+2

  else x := x+1; y := y+1;

  endif

endwhile
```



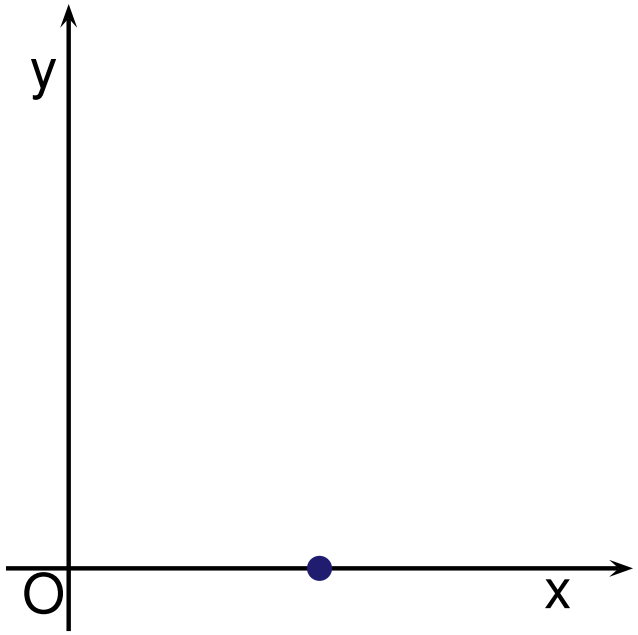
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{x = 0, y = 0\}$
```
  read(b);
  if b then x := x+2

  else x := x+1; y := y+1;

  endif

endwhile
```



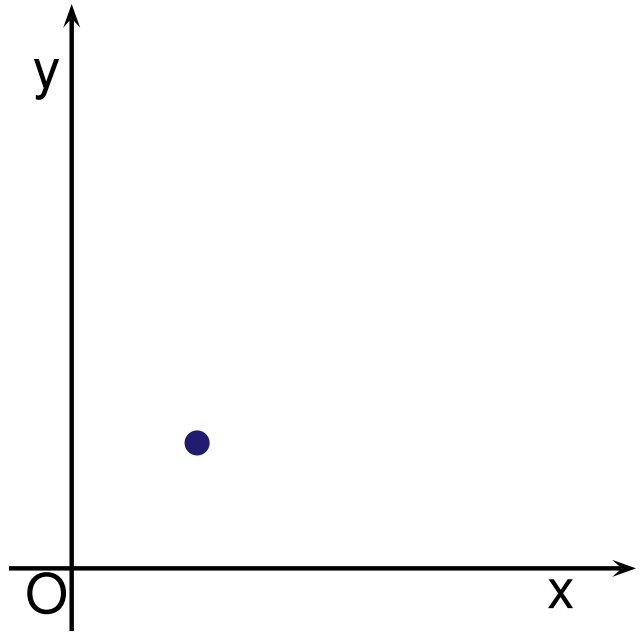
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{x = 0, y = 0\}$
```
  read(b);
  if b then x := x+2
```
$\{x = 2, y = 0\}$
```
  else x := x+1; y := y+1;

  endif

endwhile
```



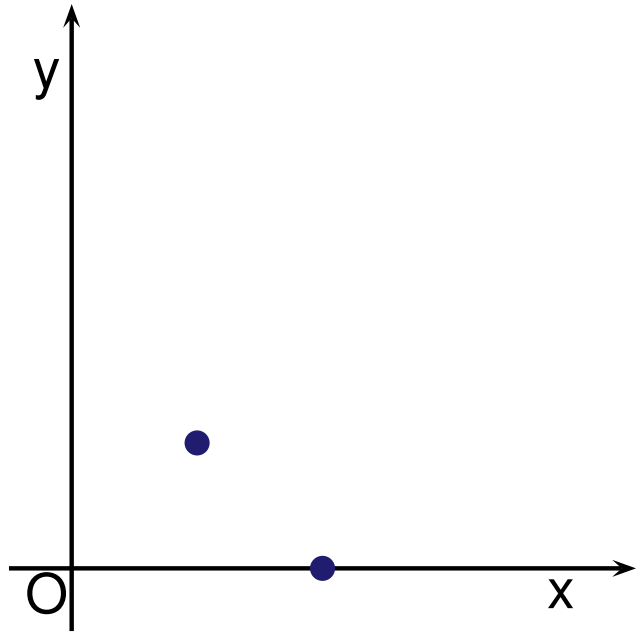
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{x = 0, y = 0\}$
```
  read(b);
  if b then x := x+2
```
$\{x = 2, y = 0\}$
```
  else x := x+1; y := y+1;
```
$\{x = 1, y = 1\}$
```
  endif

endwhile
```



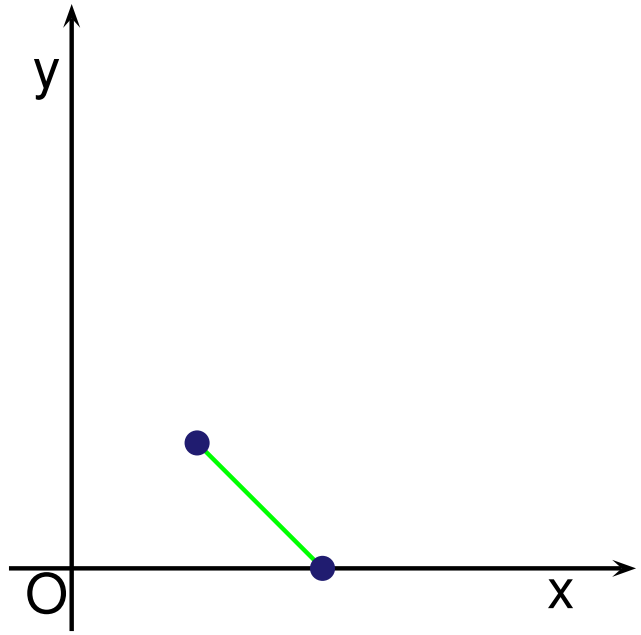
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{x = 0, y = 0\}$
```
  read(b);
  if b then x := x+2
```
$\{x = 2, y = 0\}$
```
  else x := x+1; y := y+1;
```
$\{x = 1, y = 1\}$
```
  endif
```
$\{x = 2, y = 0\} \uplus \{x = 1, y = 1\}$
```
endwhile
```



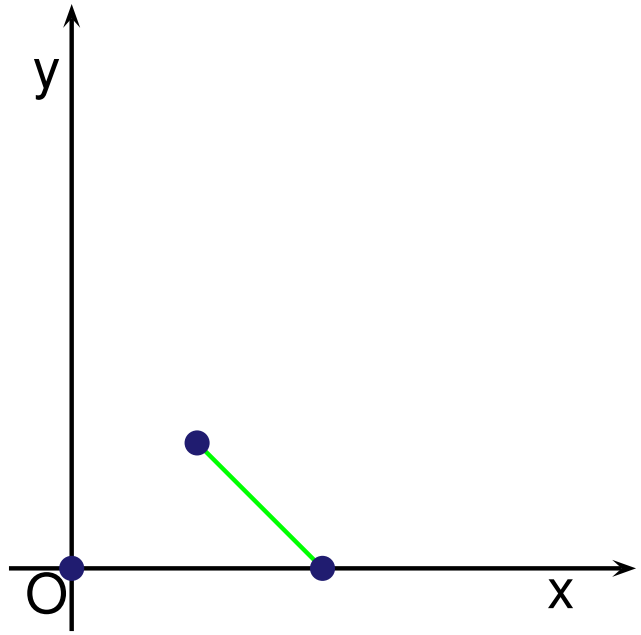
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{x = 0, y = 0\}$
```
  read(b);
  if b then x := x+2
```
$\{x = 2, y = 0\}$
```
  else x := x+1; y := y+1;
```
$\{x = 1, y = 1\}$
```
  endif
```
$\{1 \leq x \leq 2, x + y = 2\}$
```
endwhile
```



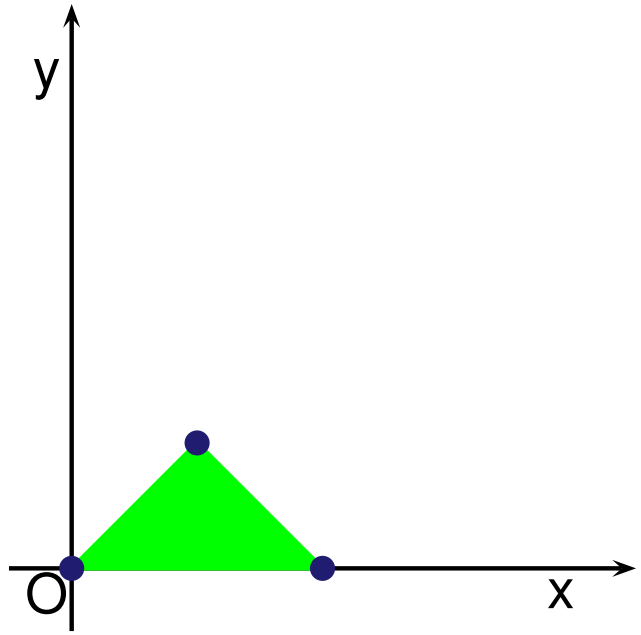
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{x = 0, y = 0\}$
$\uplus \{1 \leq x \leq 2, x + y = 2\}$
```
  read(b);
  if b then x := x+2
```
$\{x = 2, y = 0\}$
```
  else x := x+1; y := y+1;
```
$\{x = 1, y = 1\}$
```
  endif
```
$\{1 \leq x \leq 2, x + y = 2\}$
```
endwhile
```



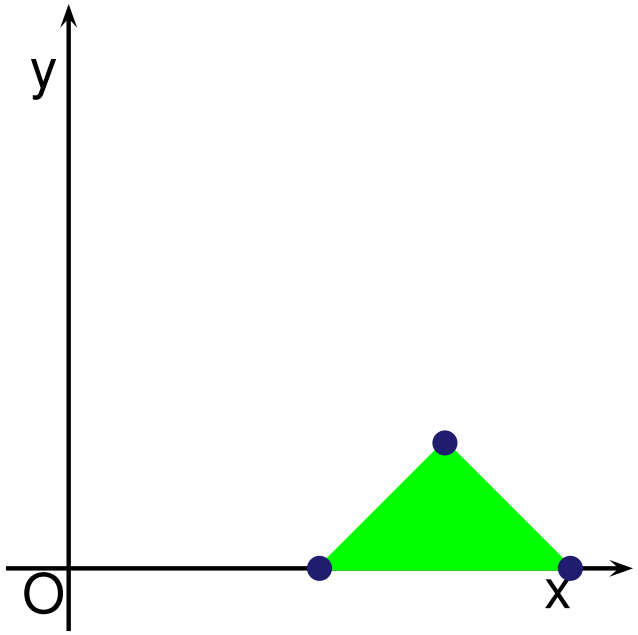
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x, x + y \leq 2\}$
```
  read(b);
  if b then x := x+2
```
$\{x = 2, y = 0\}$
```
  else x := x+1; y := y+1;
```
$\{x = 1, y = 1\}$
```
  endif
```
$\{1 \leq x \leq 2, x + y = 2\}$
```
endwhile
```



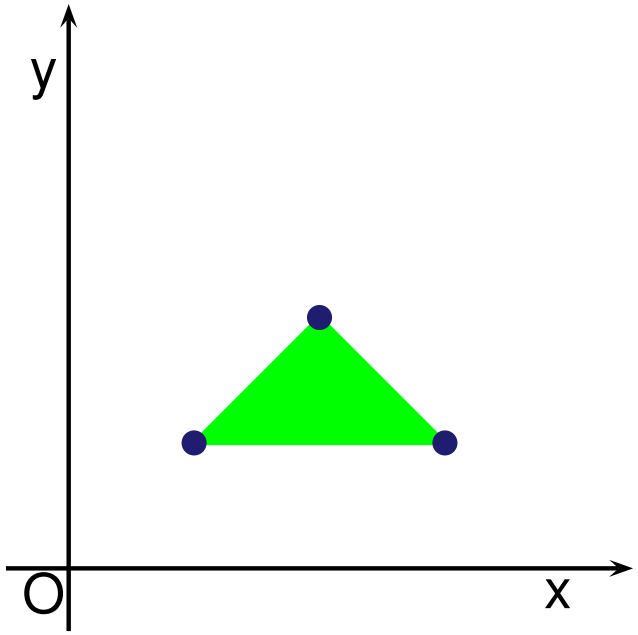
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x, x + y \leq 2\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$
```
  else x := x+1; y := y+1;
```
$\{x = 1, y = 1\}$
```
  endif
```
$\{1 \leq x \leq 2, x + y = 2\}$
```
endwhile
```



Example: The Abstract Meaning

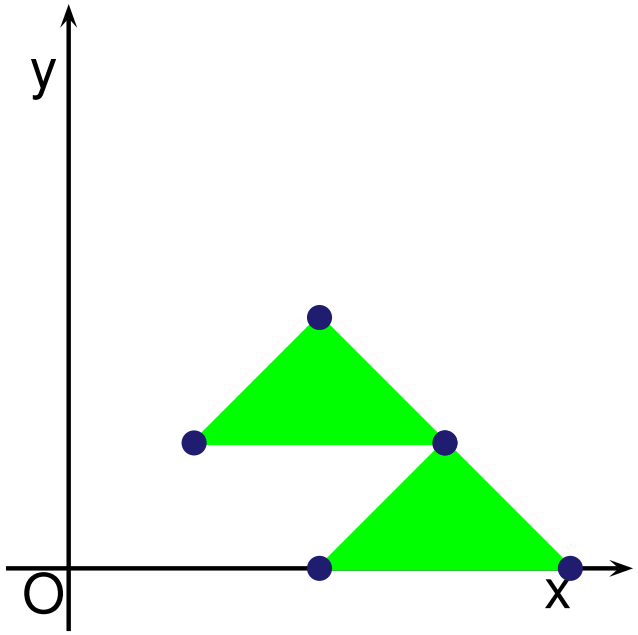
```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x, x + y \leq 2\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x, x + y \leq 4\}$
```
  endif
```
$\{1 \leq x \leq 2, x + y = 2\}$
```
endwhile
```



Example: The Abstract Meaning

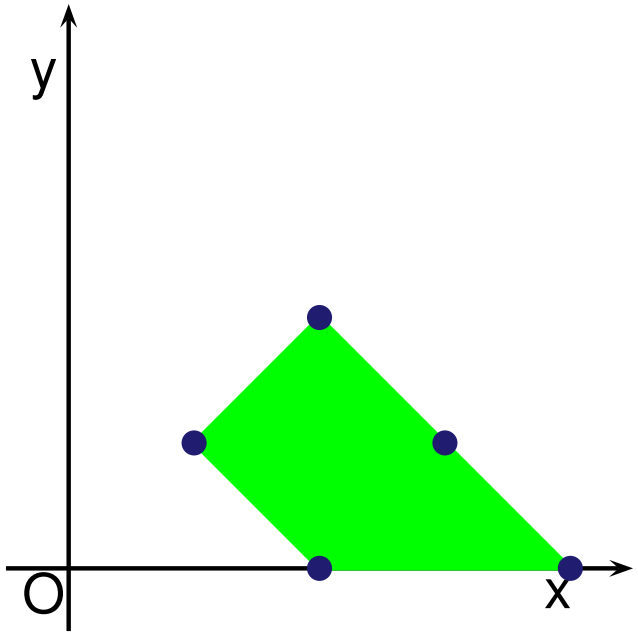
```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x, x + y \leq 2\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x, x + y \leq 4\}$
```
  endif
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$

$\uplus \{1 \leq x \leq 2, x + y = 2\}$
```
endwhile
```



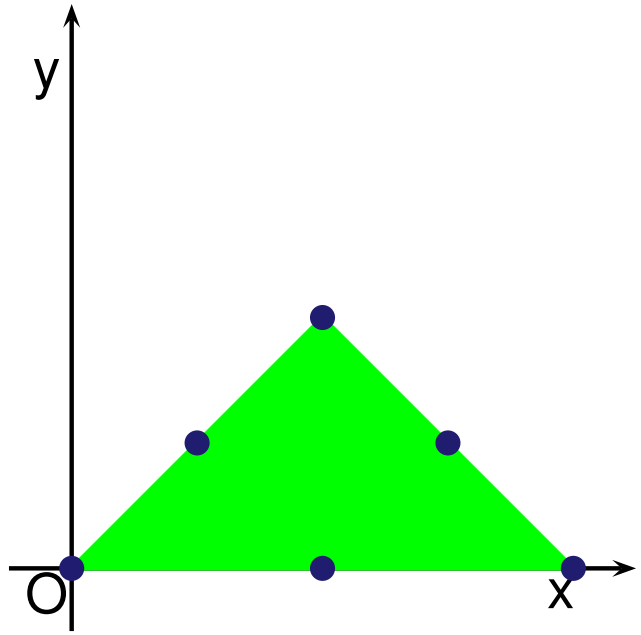
Example: The Abstract Meaning

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x, x + y \leq 2\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x, x + y \leq 4\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y \leq 4\}$
```
endwhile
```



EXAMPLE: . . . AND SO ON . . . ?

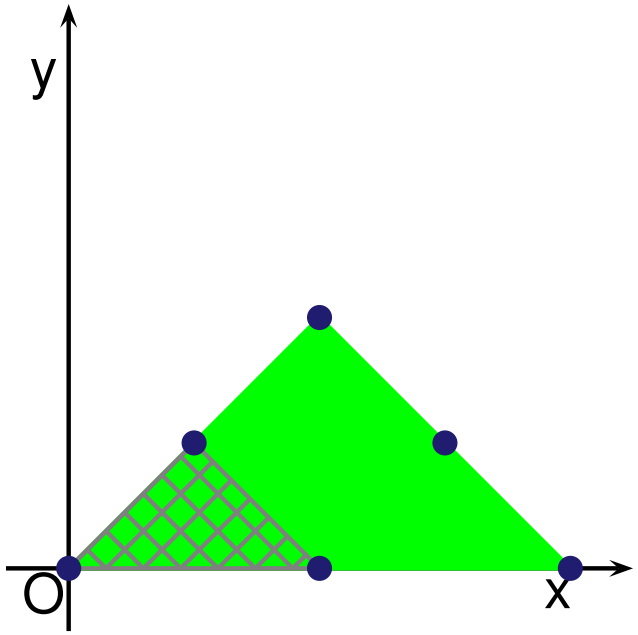
```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x, x + y \leq 4\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x, x + y \leq 4\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y \leq 4\}$
```
endwhile
```



Example: Finite Convergence using Widening

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x, x + y \leq 2\}$

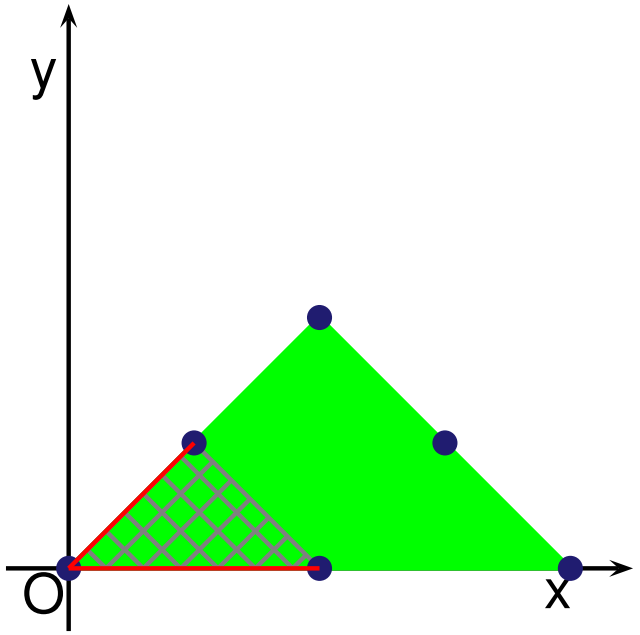
$\nabla \{0 \leq y \leq x, x + y \leq 4\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x, x + y \leq 4\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y \leq 4\}$
```
endwhile
```



Example: Finite Convergence using Widening

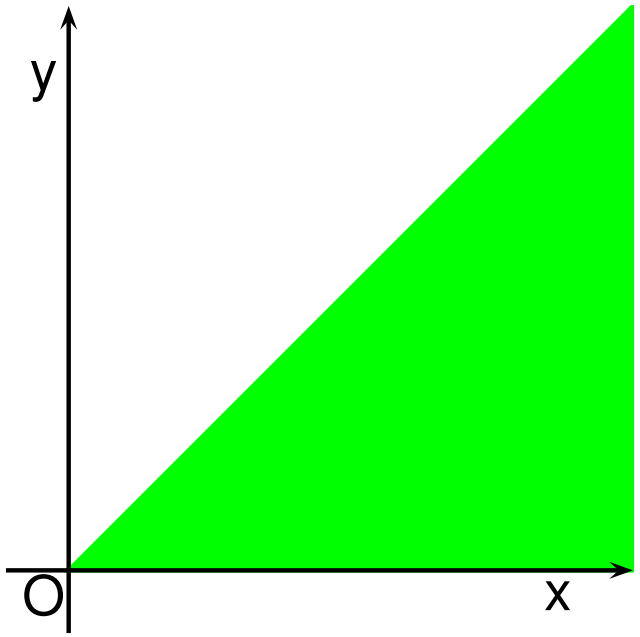
```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x, x + y \leq 2\}$

$\nabla \{0 \leq y \leq x, x + y \leq 4\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x, x + y \leq 4\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y \leq 4\}$
```
endwhile
```



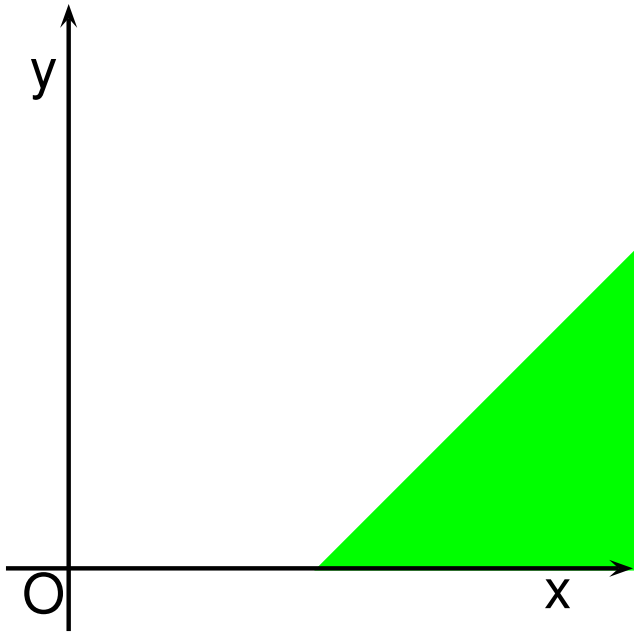
Example: An Abstract Post-Fixpoint

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2, x + y \leq 4\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x, x + y \leq 4\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y \leq 4\}$
```
endwhile
```



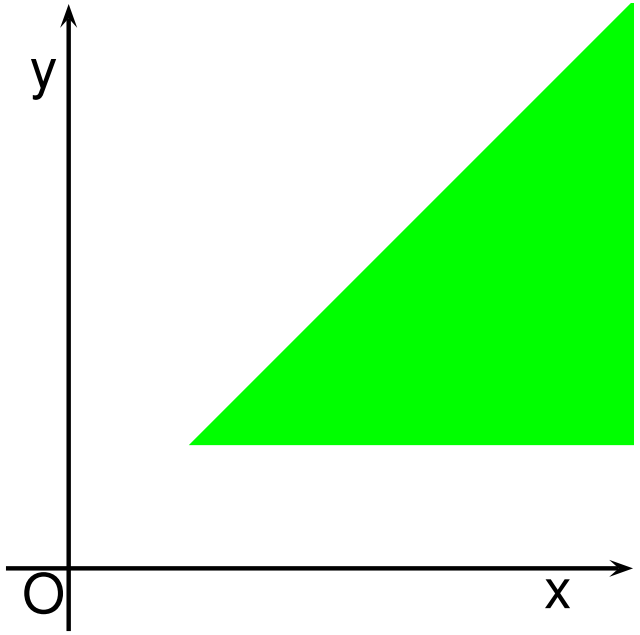
Example: Abstract Downward Iteration

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x, x + y \leq 4\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y \leq 4\}$
```
endwhile
```



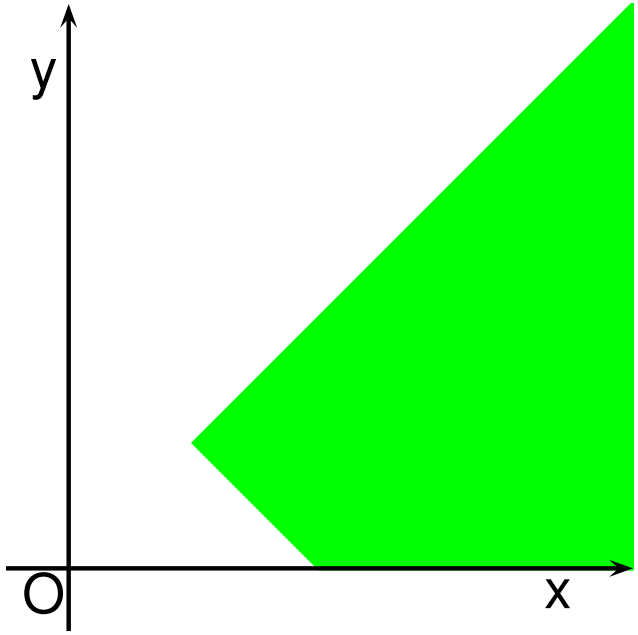
Example: Abstract Downward Iteration

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y \leq 4\}$
```
endwhile
```



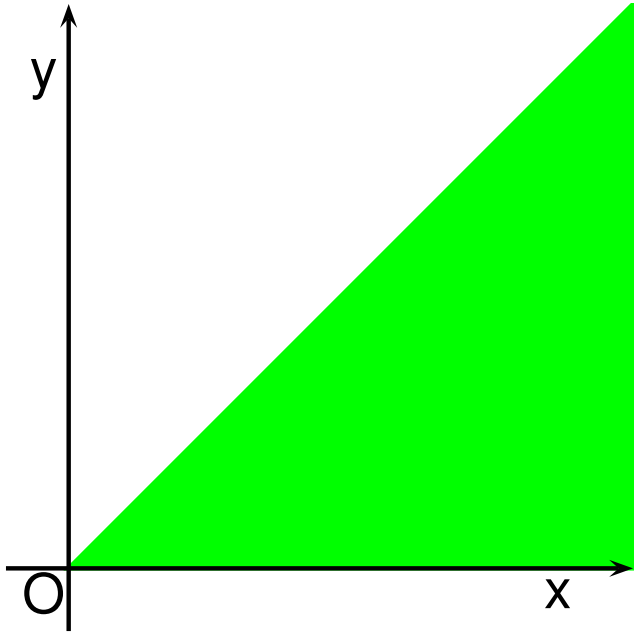
Example: Abstract Downward Iteration

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y\}$
```
endwhile
```



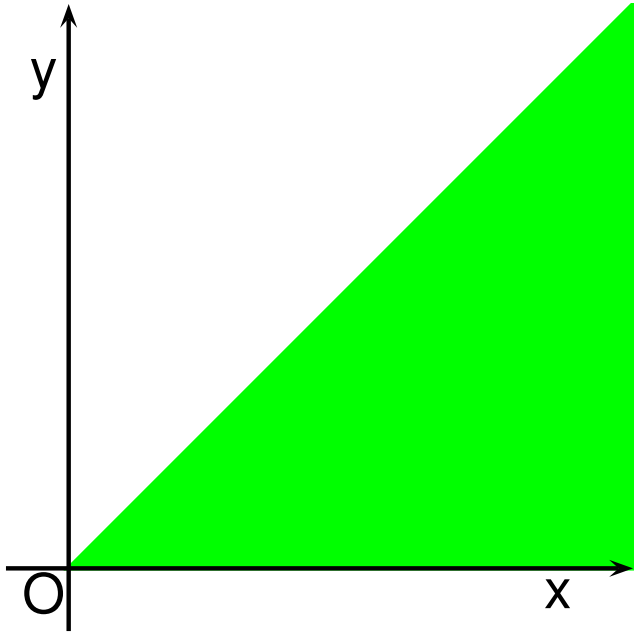
Example: Abstract Downward Iteration

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x\}$
$\cap \{0 \leq y \leq x \leq 100\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y\}$
```
endwhile
```



Example: Abstract Fixpoint

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x \leq 100\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x\}$
```
  endif
```
$\{0 \leq y \leq x, 2 \leq x + y\}$
```
endwhile
```



Example: Abstract Fixpoint

```
x := 0; y := 0;
```
$\{x = 0, y = 0\}$
```
while x <= 100 do
```
$\{0 \leq y \leq x \leq 100\}$
```
  read(b);
  if b then x := x+2
```
$\{0 \leq y \leq x - 2 \leq 100\}$
```
  else x := x+1; y := y+1;
```
$\{1 \leq y \leq x \leq 101\}$
```
  endif
```
$\{0 \leq y \leq x \leq 102, 2 \leq x + y \leq 202\}$
```
endwhile
```
$\{100 < x \leq 102, 0 \leq y \leq x, x + y \leq 202\}$

OPERATORS MAPPING

- ➜ conditional (affine) guards $\Rightarrow$ (affine) constraints
- ➜ (affine) assignments $\Rightarrow$ (affine) images
- ➜ adding variables $\Rightarrow$ adding dimensions
- ➜ removing variables $\Rightarrow$ projecting dimensions
- ➜ merging control flows $\Rightarrow$ (approximating) unions
- ➜ iteration to fixpoint $\Rightarrow$ widening + check for containment (+ narrowing)
- ➜ . . . there are others

Implementing Convex Polyhedra

- ➜ Single Description approach
  - Keeping only constraint representation
  - Fourier-Motzkin elimination procedure
  - Main issue: removal of redundant constraints
  - Well suited for few, sparse constraints
- ➜ Double Description approach
  - Keeping both constraints and generators
  - Chernikova conversion algorithm
  - Main issue: size of representations
  - Simpler redundancy elimination

The Double Description Method

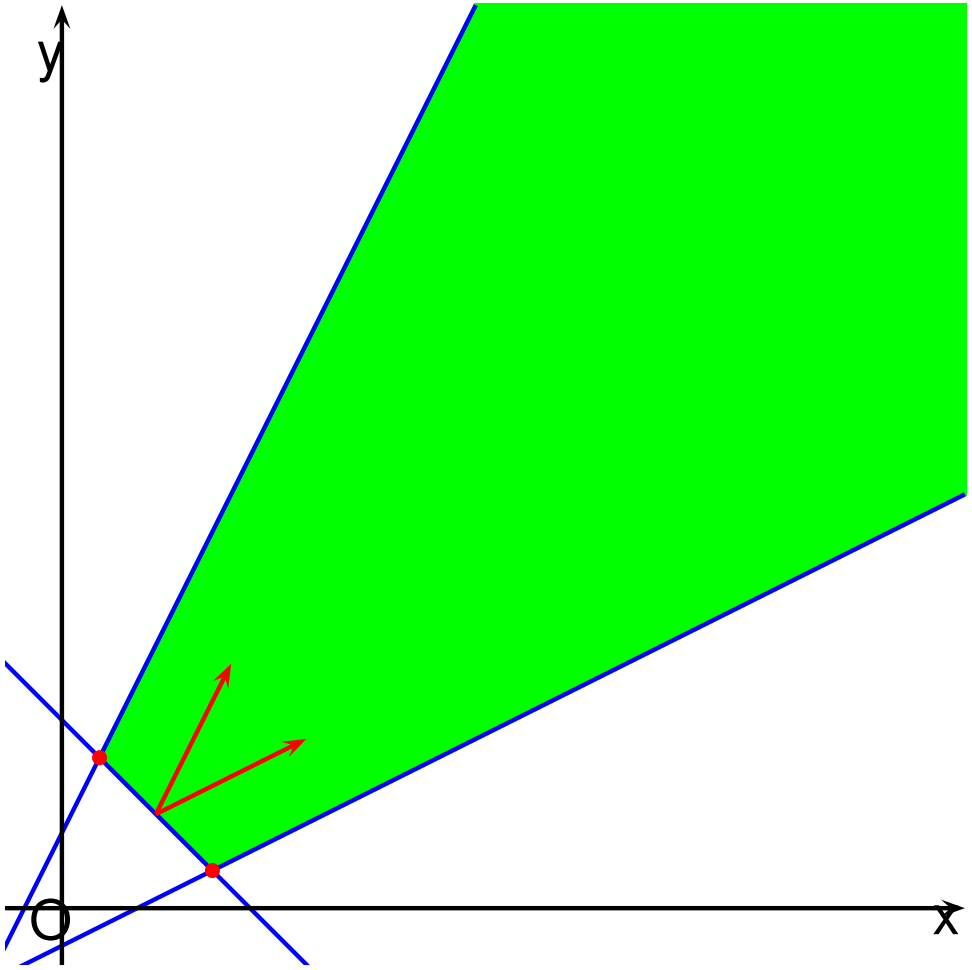
Constraint Representation

➜ $\left\{ \boldsymbol{x} \in \mathbb{R}^n \;\middle|\; A\boldsymbol{x} \leq \boldsymbol{b} \right\}$

Generator representation

➜ $\left\{ R\boldsymbol{\rho} + P\boldsymbol{\pi} \in \mathbb{R}^n \;\middle|\; \rho_i, \pi_i \geq 0, \sum \pi_i = 1 \right\}$

EXAMPLE: DOUBLE DESCRIPTION



$$\begin{cases} x + y \geq 5 \\ x - 2y \leq 2 \\ y - 2x \leq 2 \end{cases}$$

$$\begin{cases} \text{points: } \{(4,1),(1,4)\} \\ \text{rays: } \{(1,2),(2,1)\} \end{cases}$$

NOT NECESSARILY CLOSED (NNC) POLYHEDRA

Constraint Representation: strict inequalities

➜ $\left\{ \boldsymbol{x} \in \mathbb{R}^n \;\middle|\; A\boldsymbol{x} \leq \boldsymbol{b}, A'\boldsymbol{x} < \boldsymbol{b}' \right\}$

Generator representation: closure points

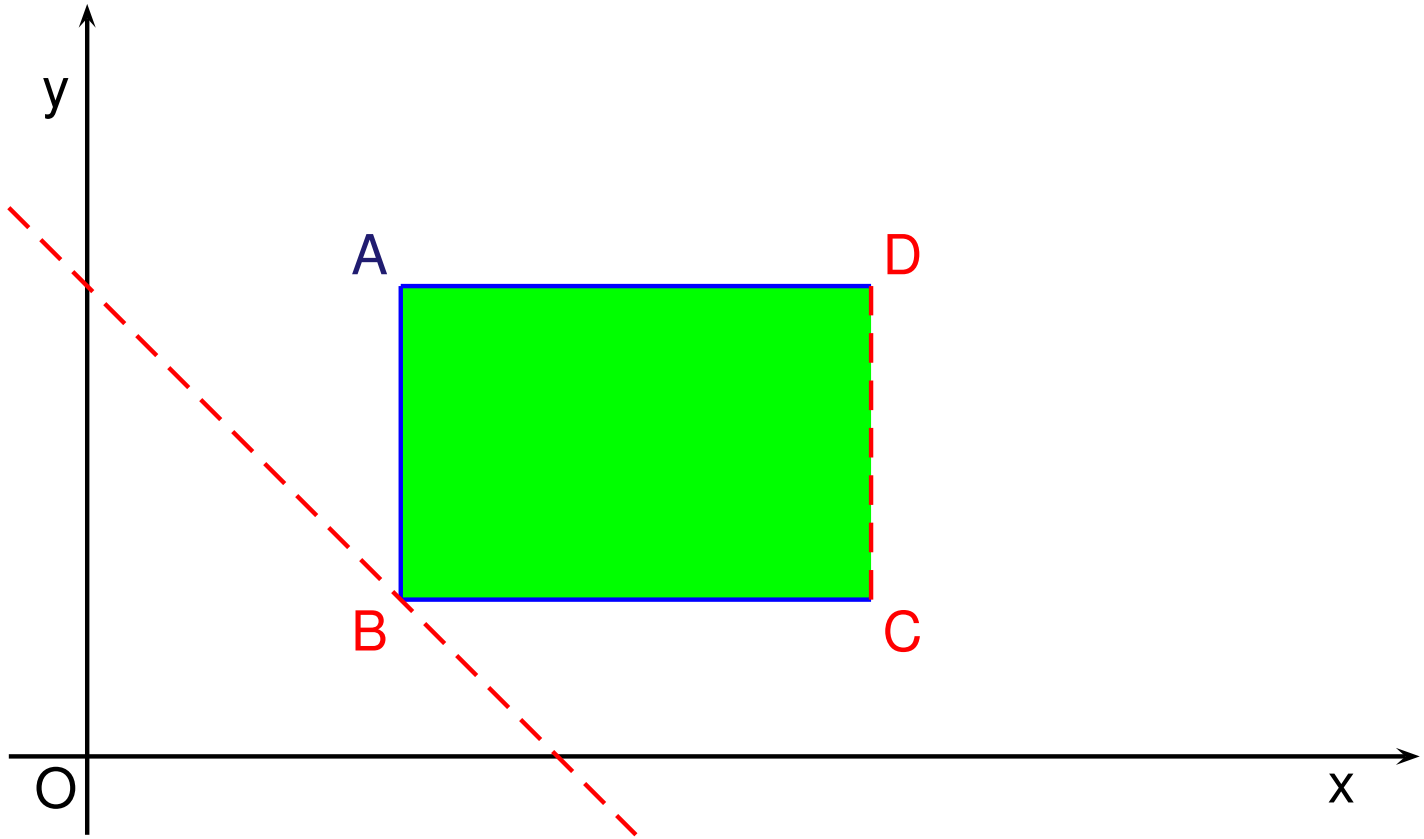
➜ $\left\{ R\boldsymbol{\rho} + P\boldsymbol{\pi} + C\boldsymbol{\gamma} \in \mathbb{R}^n \;\middle|\; \rho_i, \pi_i, \gamma_i \geq 0, \sum \pi_i + \sum \gamma_i = 1, \boldsymbol{\pi} \neq \mathbf{0} \right\}$

N. Halbwachs, Y.E. Proy, P. Raymond
Verification of Linear Hybrid Systems by Means of Convex Approximations,
SAS 1994

R. Bagnara, P.M. Hill, E. Zaffanella
Not necessarily closed convex polyhedra and the double description method
FAC 2005

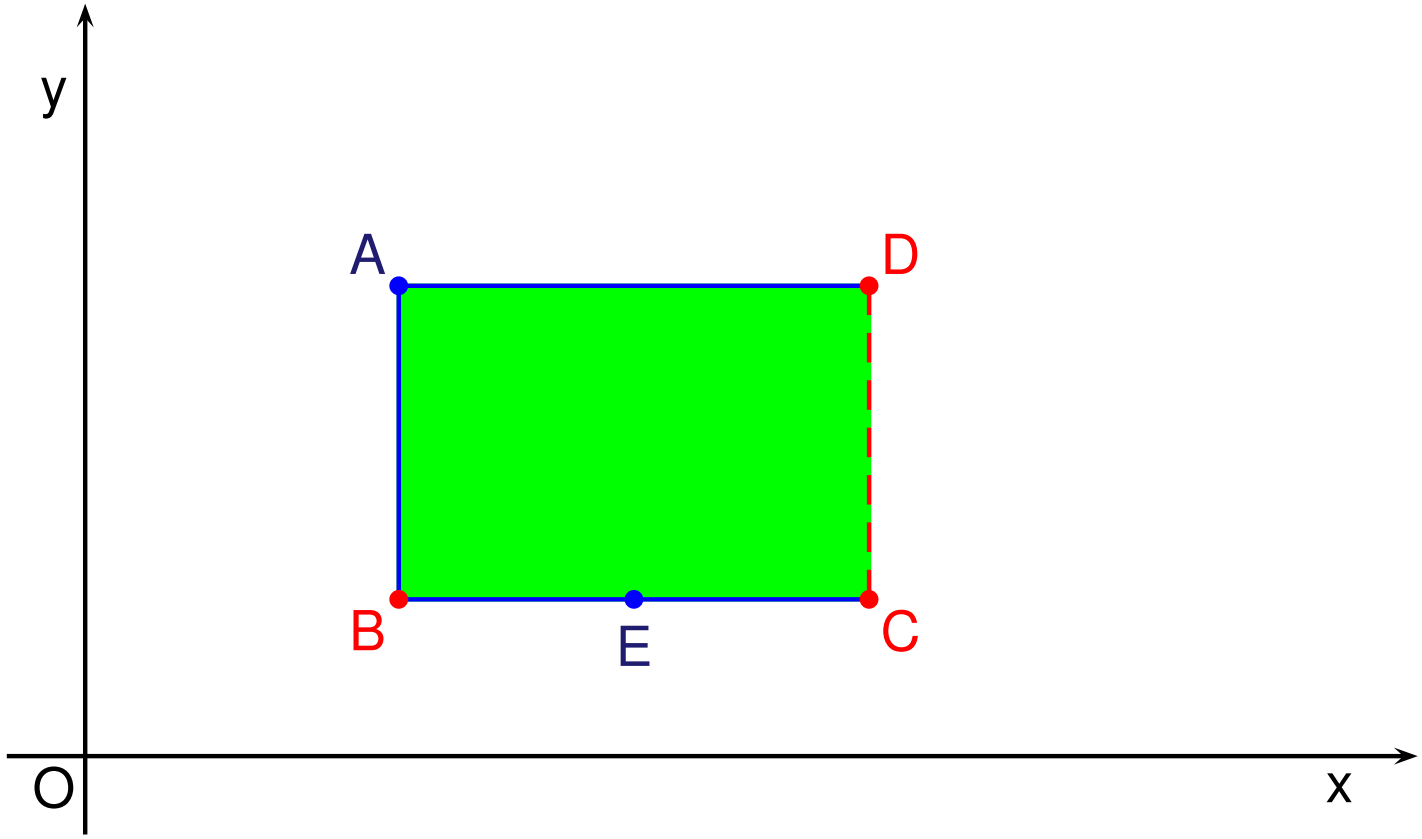
EXAMPLE OF NNC POLYHEDRON: CONSTRAINTS

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



Example of NNC Polyhedron: Generators

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



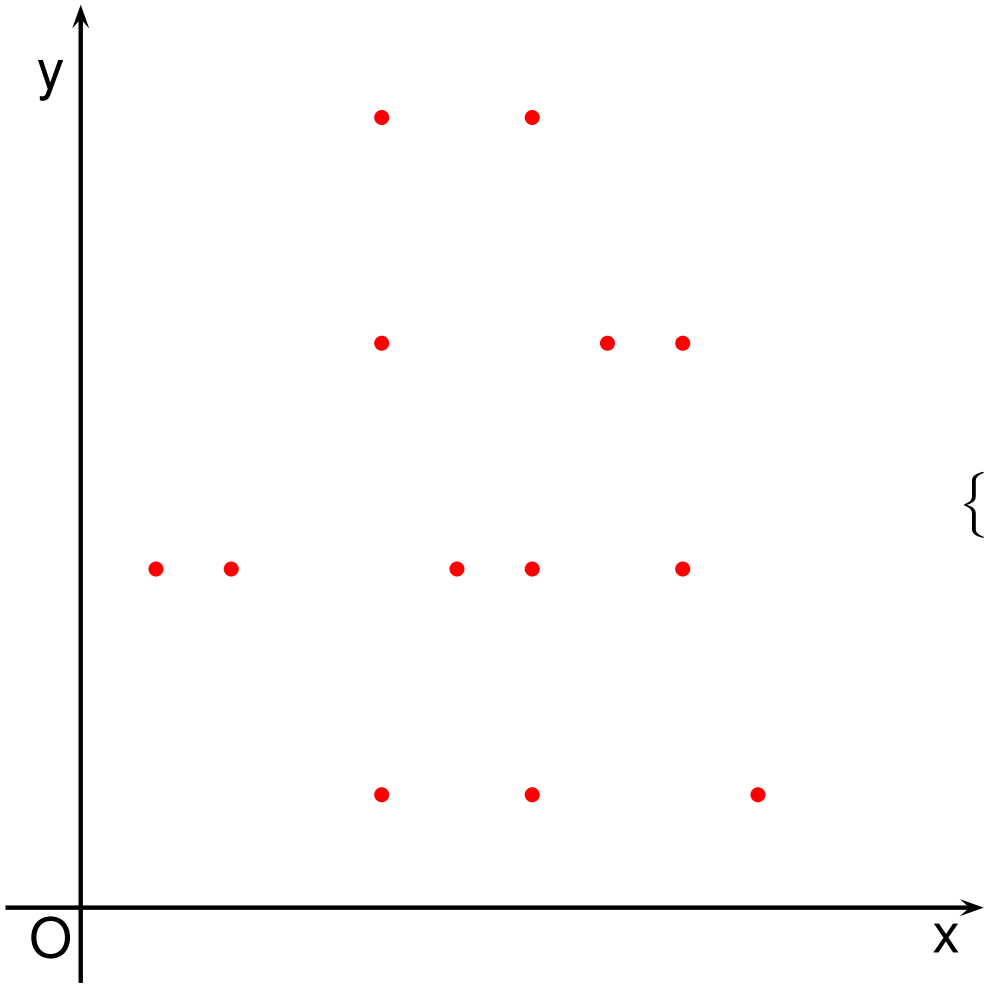
Convex Polyhedra: Pros and Cons

- ➜ Pros
  - High precision (many optimal operators)
  - Reasonable efficiency in several contexts (not everywhere)
- ➜ Cons
  - Exponential complexity bites (in some contexts)
  - Sometimes precision is not enough

Simpler Abstract Domains

Convex polyhedra are not the only option . . .

No Abstraction



$$\{\ldots,(2,9),\ldots,(12,21)\ldots,\}$$

Convex Polyhedra (I)

$$A\boldsymbol{x} \leq \boldsymbol{b}$$

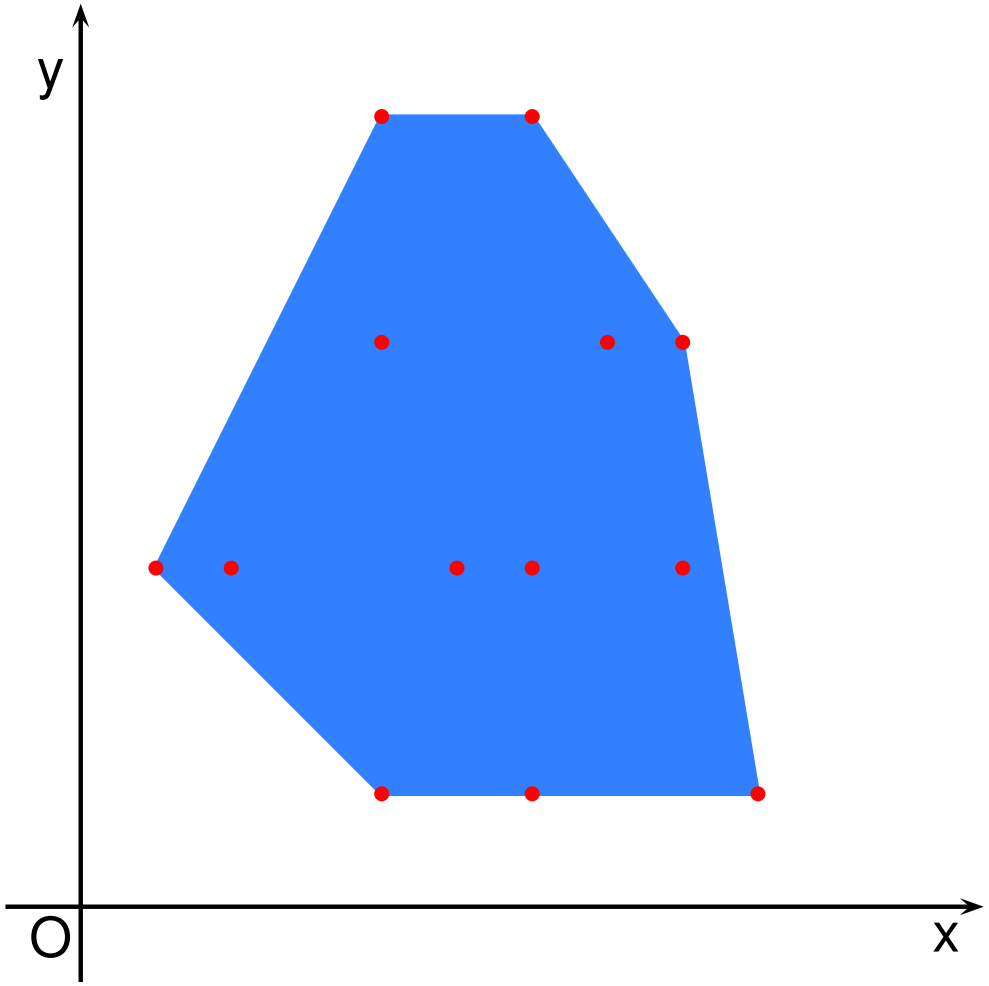
$$a_{ij}, b_j \in \mathbb{Q}$$

P. Cousot, N. Halbwachs

Automatic discovery of linear restraints among variables of a program

POPL 1978

Convex Polyhedra (II)



$$\begin{cases} 6x + y \leq 111 \\ 3x + 2y \leq 78 \\ x + y \geq 11 \\ 2x - y \geq -5 \\ y \geq 3 \\ y \leq 21 \end{cases}$$

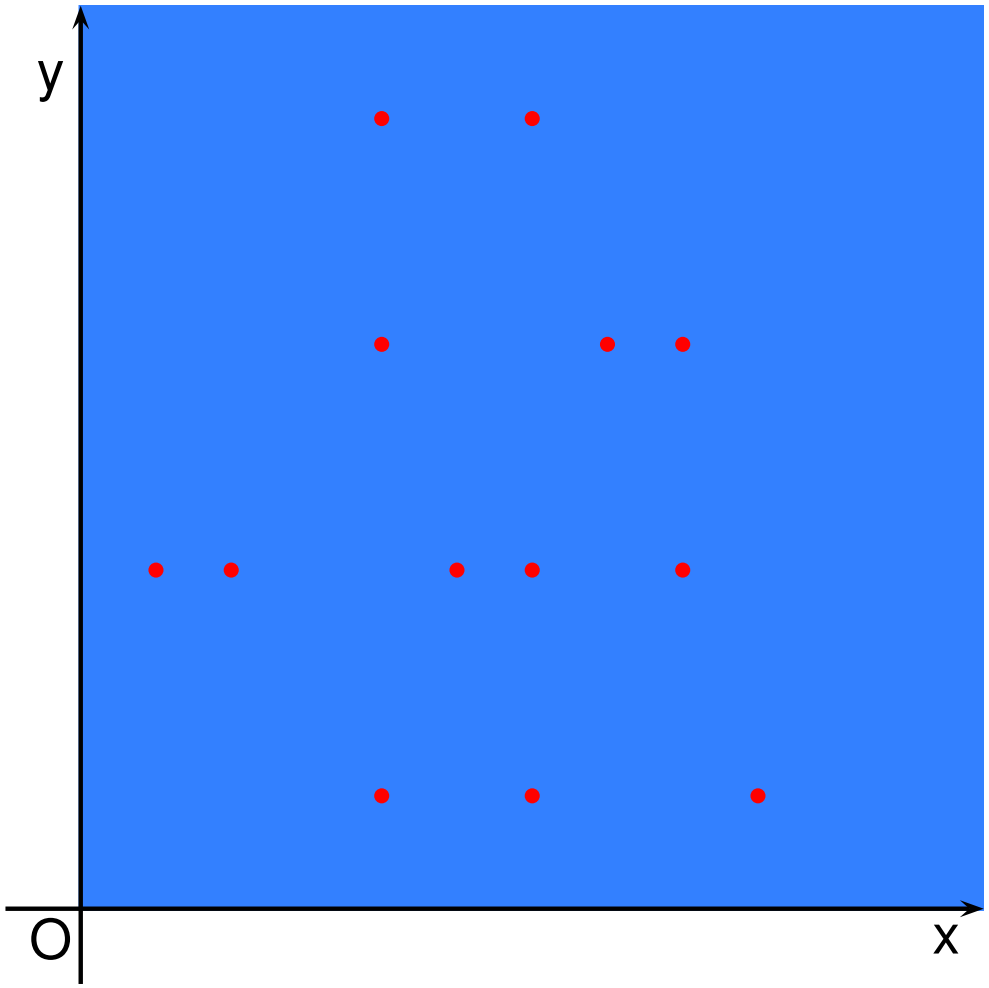
Signs (I)

$x_i \bowtie 0$

$\bowtie \in \{ \leq, =, \geq \}$

P. Cousot, R. Cousot
Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints
POPL 1977

SIGNS (II)



$$\begin{cases} x \geq 0 \\ y \geq 0 \end{cases}$$

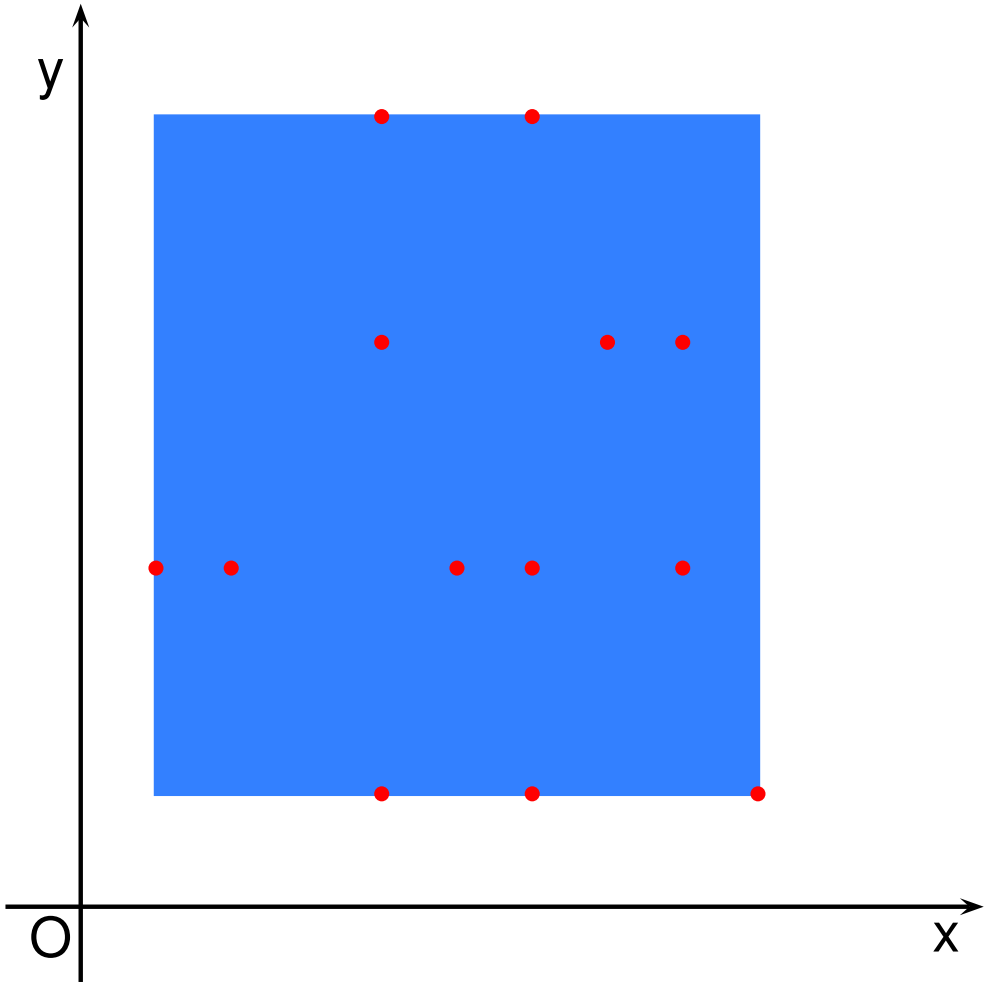
Bounding Boxes (I)

$$\boldsymbol{\ell} \leq \boldsymbol{x} \leq \boldsymbol{u}$$

$$\ell_i, u_i \in \mathbb{Q} \cup \{-\infty, +\infty\}$$

P. Cousot, R. Cousot
Static determination of dynamic properties of programs
ISOP 1976

Bounding Boxes (II)



$$\begin{cases} 2 \leq x \leq 18 \\ 3 \leq y \leq 21 \end{cases}$$

Bounded Differences (I)

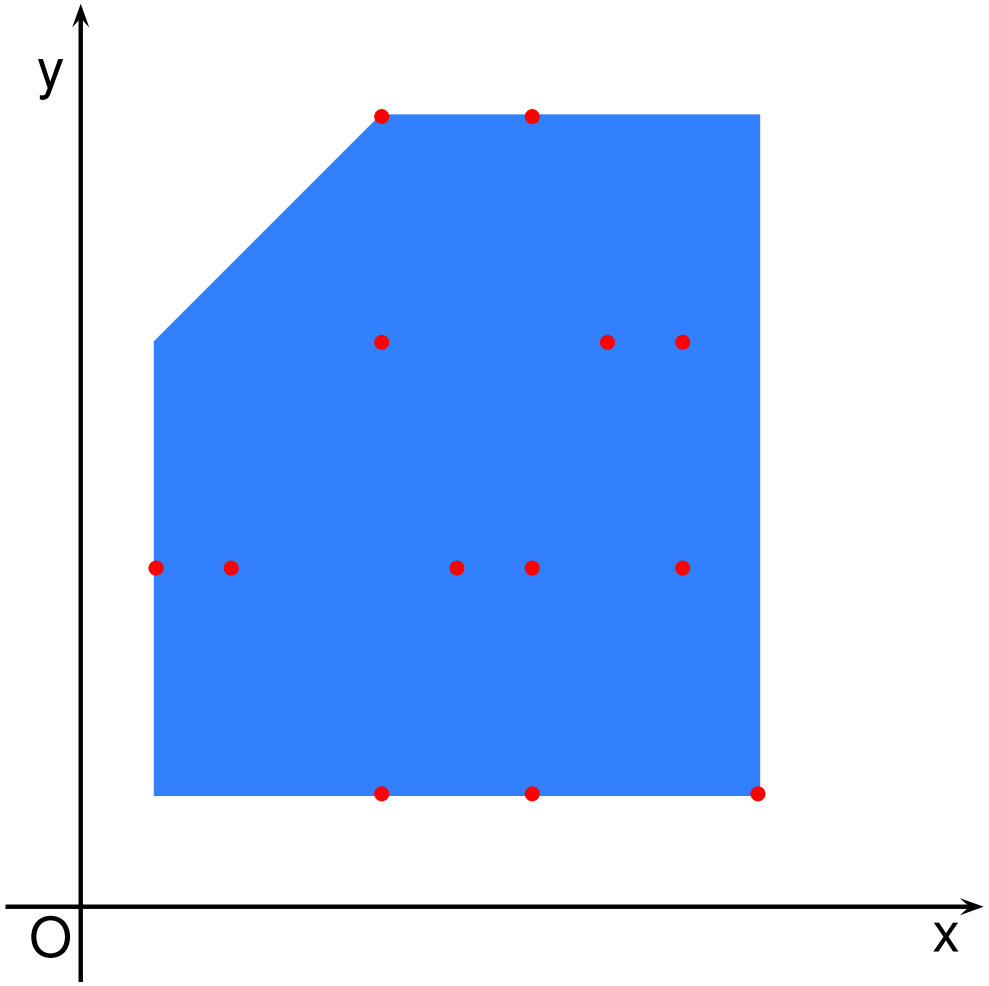
$$\ell_i \leq x_i - x_j \leq u_i$$

$$\ell_i, u_i \in \mathbb{Q} \cup \{-\infty, +\infty\}$$

R. Shaham, E.K. Kolodner, S. Sagiv
Automatic removal of array memory leaks in Java
CC2000

A. Miné
A new numerical abstract domain based on difference-bound matrices
PADO2001

Bounded Differences (II)



$$\begin{cases} 2 \leq x \leq 18 \\ 3 \leq y \leq 21 \\ -10 \leq x - y \end{cases}$$

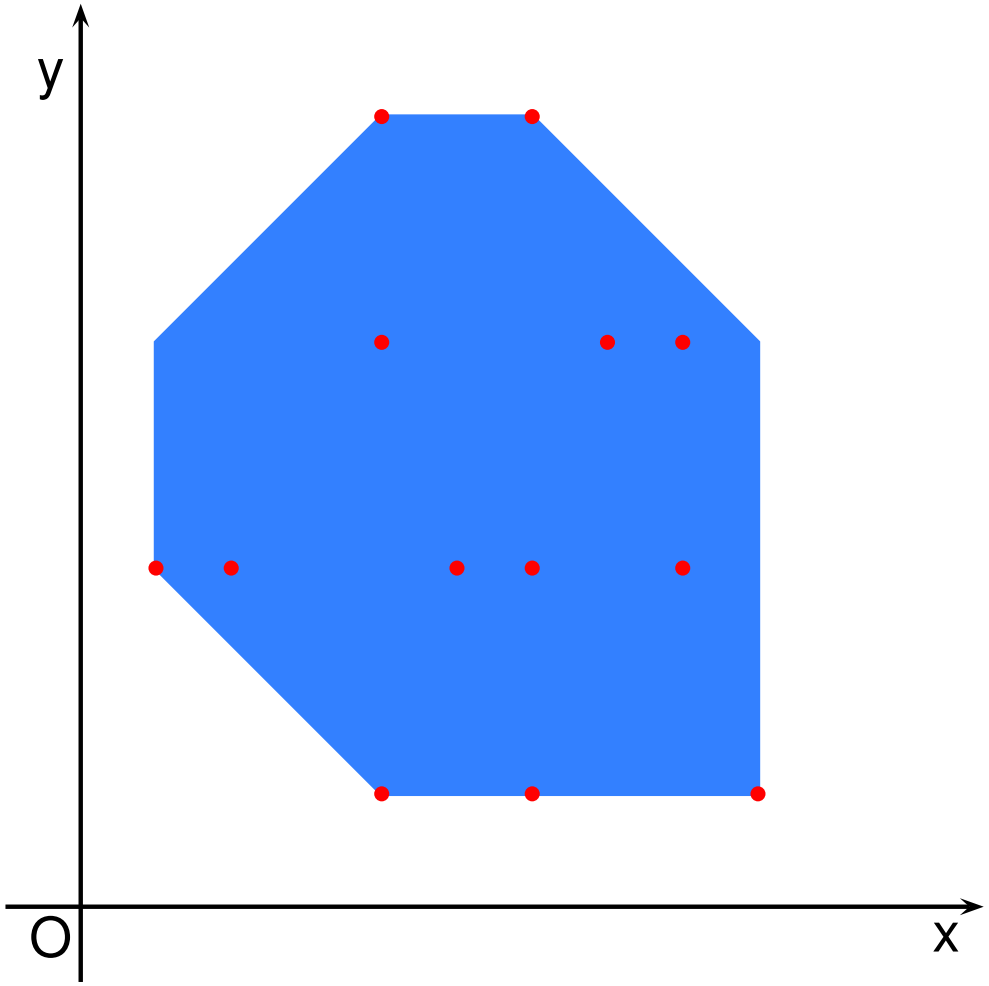
OCTAGONS (I)

$$\ell_i \leq \pm x_i \pm x_j \leq u_i$$

$$\ell_i, u_i \in \mathbb{Q} \cup \{-\infty, +\infty\}$$

A. Miné
The octagon abstract domain
WRCE 2001

Octagons (II)



$$\begin{cases} 2 \leq x \leq 18 \\ 3 \leq y \leq 21 \\ -10 \leq x - y \\ 11 \leq x + y \leq 33 \end{cases}$$

More Abstractions: Pentagons

$$\boldsymbol{\ell} \leq \boldsymbol{x} \leq \boldsymbol{u}$$

$$x_i < x_j$$

F. Logozzo, M. Fahndrich
Pentagons: A Weakly Relational Abstract Domain for the Efficient Validation of Array Accesses
SCP 2009

More Abstractions: Octahedra

$$A\boldsymbol{x} \leq \boldsymbol{b}$$

$$a_{ij} \in \{-1, 0, 1\}, b_j \in \mathbb{Q}$$

R. Clarisó, J. Cortadella
The octahedron abstract domain
SCP 2007

More Abstractions: (Bounded) Logahedra

$$a_i x_i + a_j x_j \leq b$$

$$a_i, a_j \in \{ -2^n, 0, 2^n \mid n \in \mathbb{Z}, -k \leq n \leq k \}$$

J.M. Howe, A. King
Logahedra: A New Weakly Relational Domain
ATVA 2009

Generalization: Template Polyhedra

$$A\boldsymbol{x} \leq \boldsymbol{b}$$

Note: matrix A is fixed, vector $\boldsymbol{b}$ is variable

S. Sankaranarayanan, H.B. Sipma, Z. Manna
Scalable analysis of linear systems using mathematical programming
VMCAI 2005

Affine Equalities

$$A\boldsymbol{x} = \boldsymbol{b}$$

$$a_{ij}, b_j \in \mathbb{Q}$$

M. Karr
Affine relationship among variables of a program
Acta Inf, 1976

Weighted Hexagons

$$\boldsymbol{\ell} \leq \boldsymbol{x} \leq \boldsymbol{u}$$

$$x_i \leq a \cdot x_j, (a \geq 0)$$

J. Fulara, K. Durnoga, K. Jakubczyk, A. Schubert
Relational Abstract Domain of Weighted Hexagons
ENTCS 2010

PARALLELOTOPES

$$\boldsymbol{\ell} \leq A\boldsymbol{x} \leq \boldsymbol{u}$$

Note: A (not fixed) is squared and invertible

G. Amato, F. Scozzari
The Abstract Domain of Parallelotopes
ENTCS 2012

TVPI (Two Variables per Inequality)

$$a_i x_i + a_j x_j \leq b$$

$$a_i, a_j, b \in \mathbb{Q}$$

A. Simon, A. Kind, J.M. Howe
Two Variables per Linear Inequality as an Abstract Domain
LOPSTR 2002

SUBPOLYHEDRA

$$\boldsymbol{\ell} \leq \boldsymbol{x} \leq \boldsymbol{u}$$

$$A\boldsymbol{x} + \boldsymbol{\epsilon} = \boldsymbol{b}$$

$$\boldsymbol{\epsilon} \geq \mathbf{0}$$

V. Laviron, F. Logozzo

SubPolyhedra: A (more) scalable approach to infer linear inequalities

VMCAI 2009

TOO MANY DOMAINS?

- ➜ Another handful have been proposed . . .
- ➜ Not mentioning domains that are not abstractions of polyhedra: (relational) congruences, ellipsoids, polynomial (in-) equalities, . . .
- ➜ Combinations of domains

Alternative Approaches for Efficiency

Keep Convex polyhedra but . . .

Static Variable Packing

- Statically split dimensions into smaller subsets (the packs)
- Note: packs are not necessarily disjoint
- Keep a polyhedron for each pack
- (optional) Let packs communicate to each other constraints on shared variables
- Approach widely adopted, even for weakly-relational domains

B. Blanchet, P. Cousot, R. Cousot, J. Feret,
L. Mauborgne, A. Miné, D. Monniaux, X. Rival
A static analyzer for large safety-critical software
PLDI 2003

Approximate Costly Operations

- Replace the convex polyhedral hull (a.k.a. strong join) by
  - weak join
  - inversion join

S. Sankaranarayanan, M.A. Colon, H.B. Sipma, Z. Manna
Efficient Strongly Relational Polyhedral Analysis
VMCAI 2006

A. Simon
A Note on the Inversion Join for Polyhedral Analysis
ENTCS 2010

Exploit Specific Properties

- ➜ Constraint sparsity

A. Simon, A. King
Exploiting Sparsity in Polyhedral Analysis
SAS 2005

- ➜ Dimension independence (dynamic packing)

N. Halbwachs, D. Merchat, C. Parent-Vigouroux
Cartesian Factoring of Polyhedra in Linear Relation Analysis
SAS 2003

G. Singh, M. Puschel, M. Vechev
Fast Polyhedra Abstract Domain
POPL 2017

What If Precision Is The Problem?

Precision losses typically arise:

- ➜ when merging control flow paths
  - ➜ without widenings (convex polyhedral hull)
  - ➜ with widenings
- ➜ when approximating non-linear operators

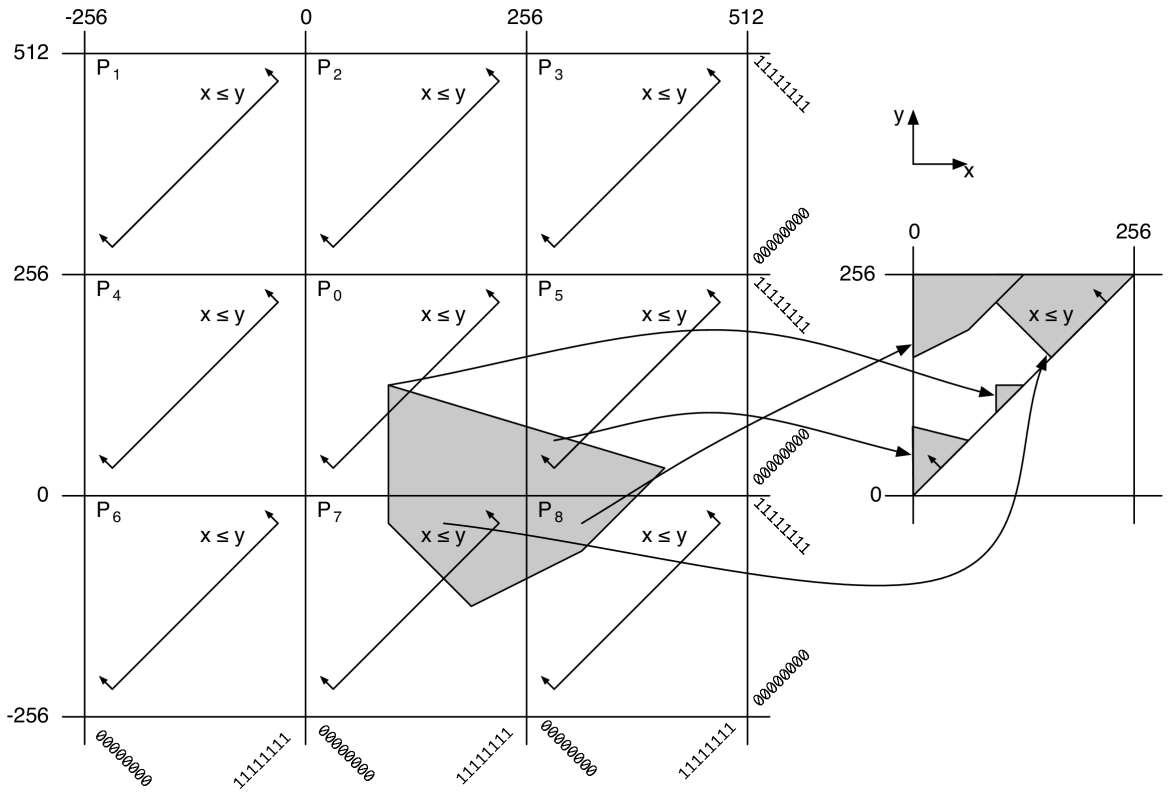
Improving the Precision of Widenings

- ➜ Many general purpose (i.e., domain independent) techniques
- ➜ Many domain specific heuristics (extrapolation operators)
- ➜ Can be combined to improve precision while enforcing termination

R. Bagnara, P.M. Hill, E. Ricci, E. Zaffanella
Precise Widening Operators for Convex Polyhedra
SAS 2003

- ➜ Can not be more precise than convex polyhedral hull

EXAMPLE: INTEGER WRAPPING (AND GUARD EVALUATION)



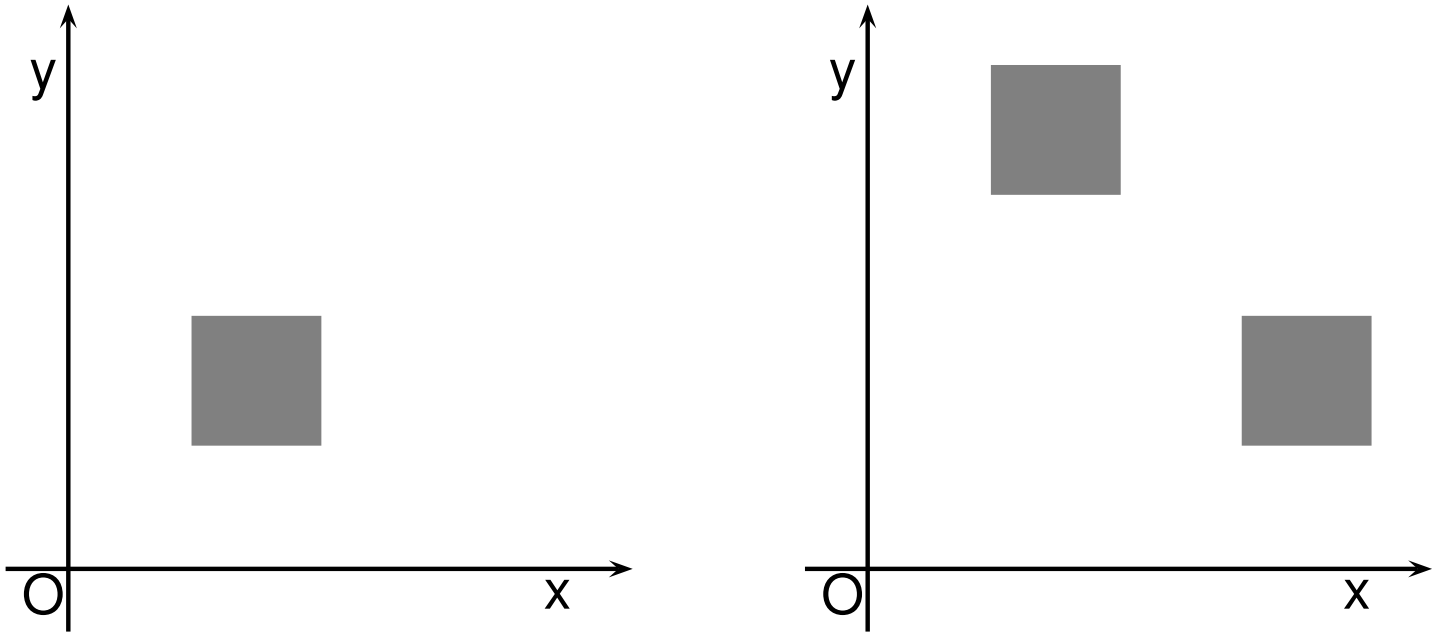
A. Simon, A. King

Taming the Wrapping of Integer Arithmetic

SAS 2004

EXAMPLE: WEAK UPDATES

```
*ptr = *ptr + 2;
```



How To Improve Convex Hulls: Avoid Them!

➜ Disjunctive domains: avoid the computation of joins

P. Cousot, R. Cousot.
Abstract interpretation frameworks
JLC 1992

➜ Trace partitioning: delay the computation of joins

X. Rival, L. Mauborgne.
The Trace Partitioning Abstract Domain
TOPLAS 2007

FINITE POWERSET DOMAINS

- ➜ Finite disjunctive sets of incomparable domain elements
- ➜ $\mathcal{F}$ additive $\Rightarrow$ apply it to each disjunct in isolation
- ➜ Careful with widening: termination guarantee is easily lost

R. Bagnara, P.M. Hill, E. Zaffanella
Widening Operators for Powerset Domains
VMCAI 2004

Detecting Exact Joins

- Different disjunctions may represent the same powerset: the fewer disjuncts, the better.
- For $\{D_1, \ldots, D_k\} \subseteq \mathbb{D}_n$, decide whether $\biguplus_{i=1}^{k} D_i = \bigcup_{i=1}^{k} D_i$.
- Too hard! But the binary case is doable: decide whether

$$D_1 \uplus D_2 = D_1 \cup D_2.$$

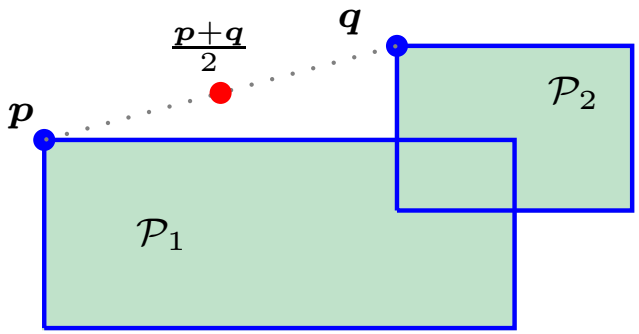
EXACT JOINS FOR CLOSED POLYHEDRA

- For non-empty $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{CP}_n$, is $\mathcal{P}_1 \uplus \mathcal{P}_2 = \mathcal{P}_1 \cup \mathcal{P}_2$?
- Problem already studied in the literature.
- Three variants considered:
  - legenda: n = dimension, l_i = ♯ cons, m_i = ♯ gens;
  - algorithm for H-polyhedra (constraint representation): $\mathrm{O}\big(l_1 l_2 \cdot \mathbf{lp}(n, l_1 + l_2)\big)$;
  - algorithm for V-polyhedra (generator representation): $\mathrm{O}\big(m_1 m_2 \cdot (\mathbf{lp}(n, m_1) + \mathbf{lp}(n, m_2))\big)$;
  - algorithm for VH-polyhedra (double description): $\mathrm{O}\big(n(l_1 + l_2)m_1 m_2\big)$;

A. Bemporad, K. Fukuda, F.D. Torrisi.
Convexity recognition of the union of polyhedra
CGTA 2001

ALGORITHM ON CLOSED (V OR VH) POLYHEDRA

➜ Intuitively, based on a mid-point checking technique.



➜ V-polyhedra: $\mathrm{O}\big(m_1 m_2 \cdot (\mathbf{lp}(n, m_1) + \mathbf{lp}(n, m_2))\big)$;

➜ VH-polyhedra: $\mathrm{O}\big(n(l_1 + l_2) m_1 m_2\big)$;

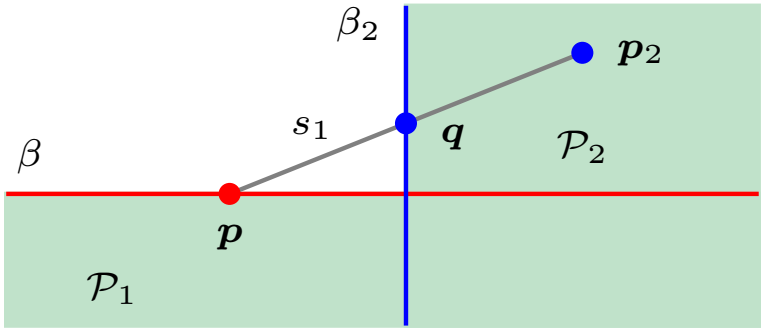
A. Bemporad, K. Fukuda, F.D. Torrisi.
Convexity recognition of the union of polyhedra
CGTA 2001

IMPROVEMENT FOR CLOSED VH-POLYHEDRA (I)

➜ Lemma: let $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{CP}_n$, non-empty

- β satisfied by $\mathcal{P}_1$, violated by $\mathcal{P}_2$
- $\boldsymbol{p} \in \mathcal{P}_1 \setminus \mathcal{P}_2$ saturates β

then $\mathcal{P}_1 \cup \mathcal{P}_2$ is not convex (hence $\mathcal{P}_1 \uplus \mathcal{P}_2 \neq \mathcal{P}_1 \cup \mathcal{P}_2$).



➜ $s_1 = (\boldsymbol{p}, \boldsymbol{q}) \in (\mathcal{P}_1 \uplus \mathcal{P}_2) \setminus (\mathcal{P}_1 \cup \mathcal{P}_2)$

Improvement for Closed VH-Polyhedra (II)

➜ Theorem leading to improved algorithm:

$\mathcal{P}_1 \uplus \mathcal{P}_2 \neq \mathcal{P}_1 \cup \mathcal{P}_2$ iff $\exists$ constraint β_1 and generator g_1 of $\mathcal{P}_1$ s.t.

① g_1 saturates β_1,

② $\mathcal{P}_2$ violates β_1,

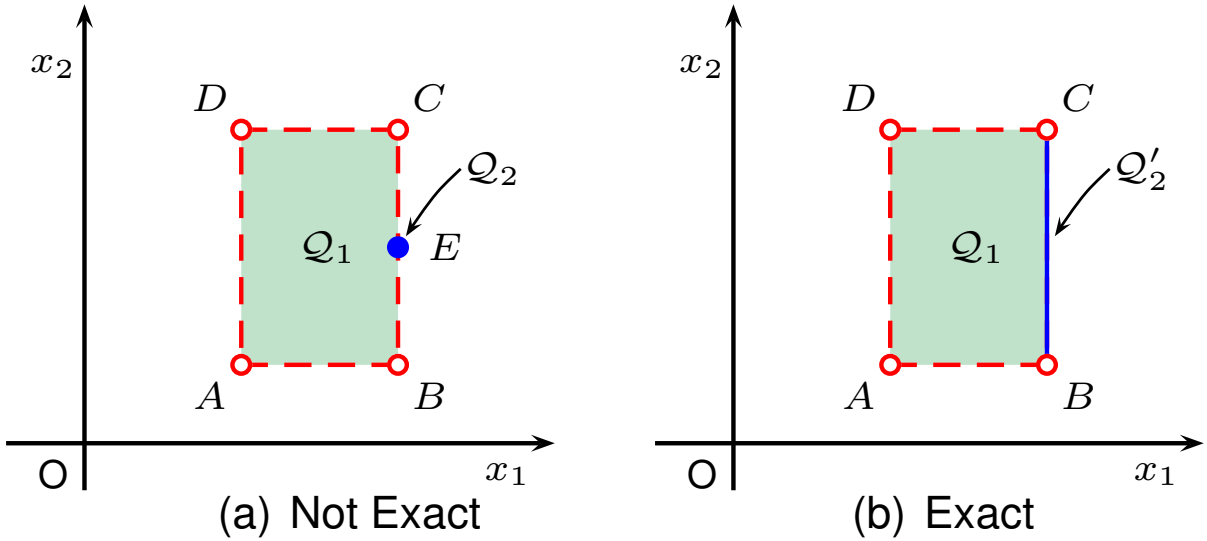
③ $\mathcal{P}_2$ does not subsume g_1.

➜ (Asymmetric) complexity bound in $\mathrm{O}\big(n(l_1 m_1 + l_1 m_2 + l_2 m_1)\big)$.

R. Bagnara, P. M. Hill, E. Zaffanella.
Exact Join Detection for Convex Polyhedra and Other Numerical Abstractions
CGTA 2010

EXACT JOINS FOR NNC POLYHEDRA (I)

- ➜ Problem never studied before by the literature
- ➜ More complex result (due to several awkward cases)
- ➜ Convexity recognition is no longer enough

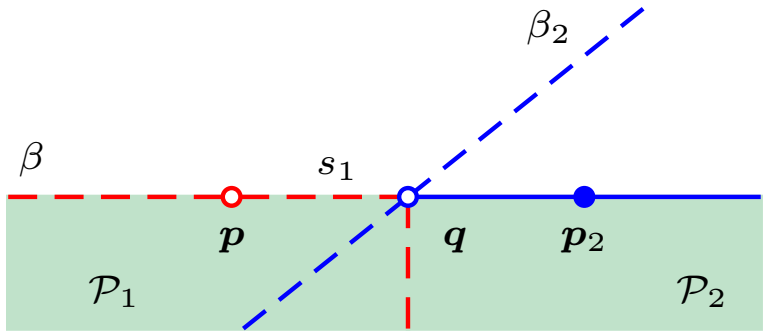


LEMMA FOR NNC VH-POLYHEDRA

➜ Lemma: let $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{P}_n$, non-empty

- β satisfied by $\mathcal{P}_1$, violated by $\mathcal{P}_2$,
- $\boldsymbol{p} \in \mathbb{C}(\mathcal{P}_1) \setminus \mathbb{C}(\mathcal{P}_2)$ saturates β

then $\mathcal{P}_1 \uplus \mathcal{P}_2 \neq \mathcal{P}_1 \cup \mathcal{P}_2$.



➜ $s_1 = (\boldsymbol{p}, \boldsymbol{q}) \in (\mathcal{P}_1 \uplus \mathcal{P}_2) \setminus (\mathcal{P}_1 \cup \mathcal{P}_2)$

AN ALGORITHM FOR NNC VH-POLYHEDRA

➜ Theorem: $\mathcal{P}_1 \uplus \mathcal{P}_2 \neq \mathcal{P}_1 \cup \mathcal{P}_2$ iff
for some $i, j \in \{1, 2\}$, $i \neq j$, there exist $g_i \in \mathcal{G}_i$ and $\beta_i \in \mathcal{C}_i$ s.t.:

1. g_i saturates β_i
2. $\mathcal{P}_j$ violates β_i
3. at least one of the following holds:

3.1. g_i is a point, β_i is non-strict and $g_i \notin \mathbb{C}(\mathcal{P}_j)$

3.2. g_i is a ray or closure point not subsumed by $\mathcal{P}_j$

3.3. β_i is strict and saturated by a point $\boldsymbol{p} \in (\mathcal{P}_1 \uplus \mathcal{P}_2) \setminus \mathcal{P}_j$

R. Bagnara, P. M. Hill, E. Zaffanella.
Exact Join Detection for Convex Polyhedra and Other Numerical Abstractions
CGTA 2010

AN ALGORITHM FOR NNC VH-POLYHEDRA (II)

➜ Theorem: $\mathcal{P}_1 \uplus \mathcal{P}_2 \neq \mathcal{P}_1 \cup \mathcal{P}_2$ iff
for some $i, j \in \{1, 2\}$, $i \neq j$, there exist $g_i \in \mathcal{G}_i$ and $\beta_i \in \mathcal{C}_i$ s.t.:

1. g_i saturates β_i
2. $\mathcal{P}_j$ violates β_i
3. at least one of the following holds:

3.1. g_i is a point, β_i is non-strict and $g_i \notin \mathbb{C}(\mathcal{P}_j)$

3.2. g_i is a ray or closure point not subsumed by $\mathcal{P}_j$

3.3. β_i is strict and saturated by a point $\boldsymbol{p} \in (\mathcal{P}_1 \uplus \mathcal{P}_2) \setminus \mathcal{P}_j$

AN ALGORITHM FOR NNC VH-POLYHEDRA (II)

➜ Theorem: $\mathcal{P}_1 \uplus \mathcal{P}_2 \neq \mathcal{P}_1 \cup \mathcal{P}_2$ iff
for some $i, j \in \{1, 2\}$, $i \neq j$, there exist $g_i \in \mathcal{G}_i$ and $\beta_i \in \mathcal{C}_i$ s.t.:

1. g_i saturates β_i
2. $\mathcal{P}_j$ violates β_i
3. at least one of the following holds:

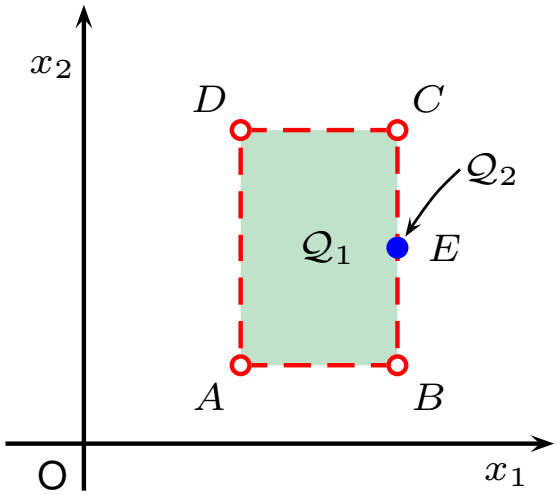
3.1. g_i is a point, β_i is non-strict and $g_i \notin \mathbb{C}(\mathcal{P}_j)$

3.2. g_i is a ray or closure point not subsumed by $\mathcal{P}_j$

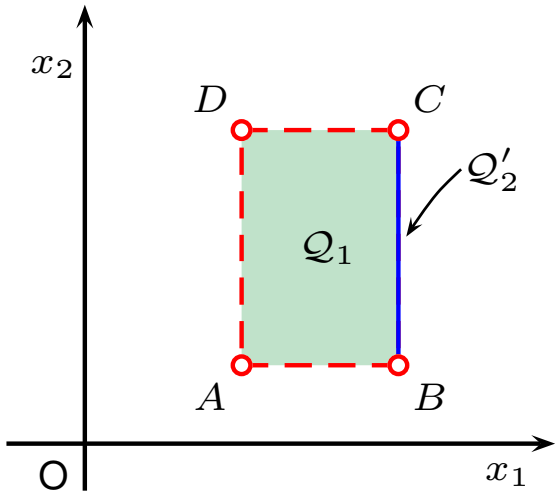
3.3. β_i is strict and saturated by a point $\boldsymbol{p} \in (\mathcal{P}_1 \uplus \mathcal{P}_2) \setminus \mathcal{P}_j$

EXACT JOINS FOR NNC POLYHEDRA: CASE 3.2

➜ $g_i = B$ is a ray or closure point of $\mathcal{P}_i = \mathcal{Q}_1$ not subsumed by $\mathcal{P}_j = \mathcal{Q}_2$



(c) Not Exact



(d) Exact

AN ALGORITHM FOR NNC VH-POLYHEDRA (III)

➜ Theorem: $\mathcal{P}_1 \uplus \mathcal{P}_2 \neq \mathcal{P}_1 \cup \mathcal{P}_2$ iff
for some $i, j \in \{1, 2\}$, $i \neq j$, there exist $g_i \in \mathcal{G}_i$ and $\beta_i \in \mathcal{C}_i$ s.t.:

1. g_i saturates β_i
2. $\mathcal{P}_j$ violates β_i
3. at least one of the following holds:

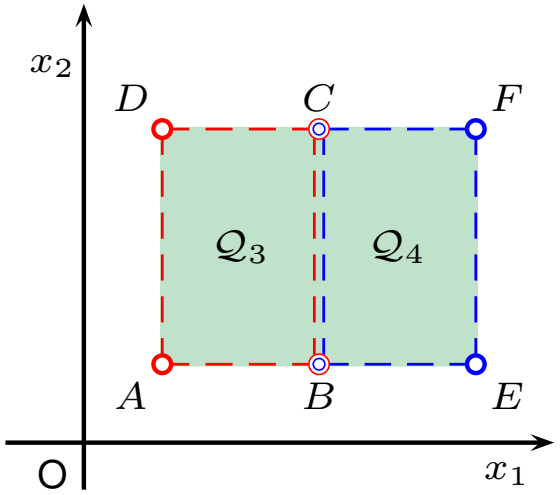
3.1. g_i is a point, β_i is non-strict and $g_i \notin \mathbb{C}(\mathcal{P}_j)$

3.2. g_i is a ray or closure point not subsumed by $\mathcal{P}_j$

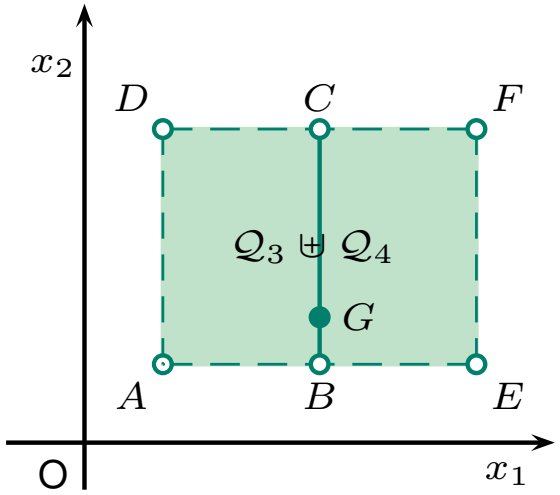
3.3. β_i is strict and saturated by a point $\boldsymbol{p} \in (\mathcal{P}_1 \uplus \mathcal{P}_2) \setminus \mathcal{P}_j$

EXACT JOINS FOR NNC POLYHEDRA: CASE 3.3

➜ $\beta_i \equiv (B, C)$ of $\mathcal{Q}_3$ is strict and saturated by point $G \in (\mathcal{Q}_3 \uplus \mathcal{Q}_4) \setminus \mathcal{Q}_4$, hence join is not exact

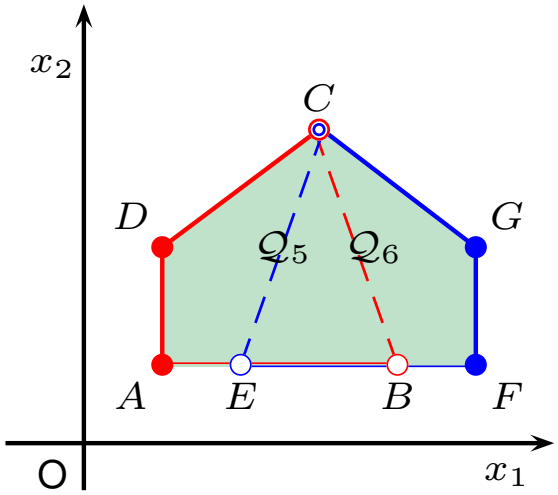


(e) Not Exact

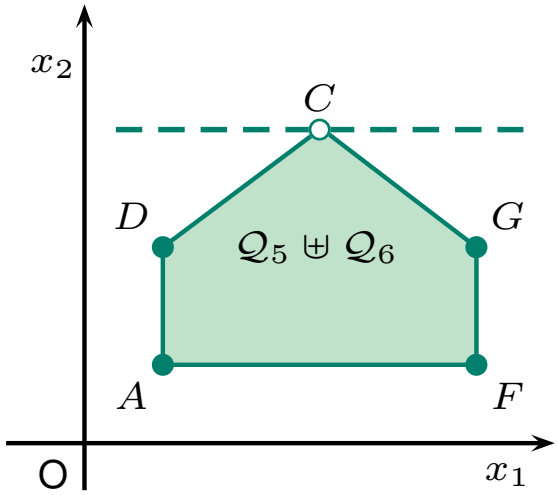


(f) Join

EXACT JOINS FOR NNC POLYHEDRA: CASE 3.3



(g) Exact



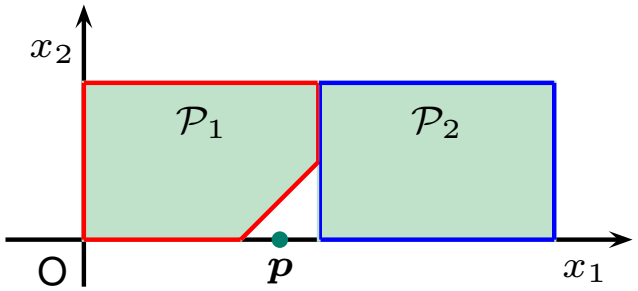
(h) Join

Exact Joins for Other Abstractions

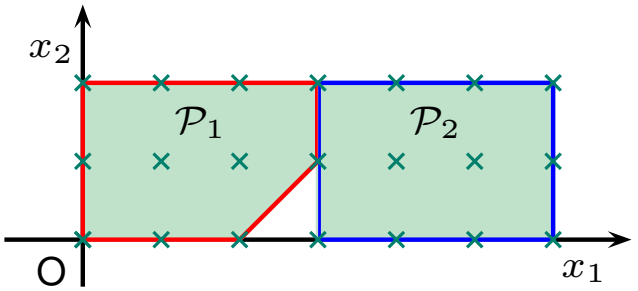
- Efficient algorithms also for:
  - attribute independent Cartesian products of simple domains such as (rational or integer) intervals, congruence equations, modulo intervals, circular linear progressions;
  - (rational or integer) bounded difference shapes;
  - (rational or integer) octagonal shapes.

R. Bagnara, P. M. Hill, E. Zaffanella.
Exact Join Detection for Convex Polyhedra and Other Numerical Abstractions
CGTA 2010

EXAMPLE: EXACT JOIN FOR BD SHAPES



(i) Rational Case: Not Exact



(j) Integer Case: Exact

Conclusion

- ➜ Polyhedral computations are very important in the field of analysis and verification of computing systems.
- ➜ The complexity/precision tradeoff is particularly severe:
  - giving up some precision is sometimes necessary
  - giving up too much precision might be undesirable
- ➜ This is why researchers came up with such a variety of abstract domains and approximate algorithms
- ➜ There still are a few open issues that need a satisfactory solution
- ➜ Tiny improvements on the theoretical side sometimes have huge effects on the practical side