Selective Unification in Constraint Logic Programming

Fred Mesnard

University of Réunion Island

Joint work with Étienne Payet (University of Réunion Island) and Germán Vidal (Technical University of Valencia)

Outline

Test-case generation for automatic software testing

Random input data:

- the most used approach
- simple, fast, sound, but poor coverage in general

Based on symbolic execution:

- replace concrete inputs by symbolic inputs, extend semantics (add a "path condition" to each state, etc)
- build a search tree, solve constraints in leaves to produce test cases
- good coverage, huge search space (incompleteness), complex constraints should be simplified (unsoundness due to abstraction)

Alternative: concolic testing

Test-case generation for automatic software testing

Random input data:

- the most used approach
- simple, fast, sound, but poor coverage in general

Based on symbolic execution:

- replace concrete inputs by symbolic inputs, extend semantics (add a "path condition" to each state, etc)
- build a search tree, solve constraints in leaves to produce test cases
- good coverage, huge search space (incompleteness), complex constraints should be simplified (unsoundness due to abstraction)

Alternative: concolic testing

Test-case generation for automatic software testing

Random input data:

- the most used approach
- simple, fast, sound, but poor coverage in general

Based on symbolic execution:

- replace concrete inputs by symbolic inputs, extend semantics (add a "path condition" to each state, etc)
- build a search tree, solve constraints in leaves to produce test cases
- good coverage, huge search space (incompleteness), complex constraints should be simplified (unsoundness due to abstraction)

Alternative: concolic testing

Concolic testing

Very popular in imperative and OO programming languages
Java PathFinder (NASA), Cute and jCute (UIUC), Klee,...

Useful for

- test case generation
- debugging
- ...

Concolic stands for concrete + symbolic execution

Concolic testing

Very popular in imperative and OO programming languages
Java PathFinder (NASA), Cute and jCute (UIUC), Klee,...

Useful for

- test case generation
- debugging
- ...

Concolic stands for concrete + symbolic execution

Concolic testing: basic idea

Let s_0 be a concrete state   Let α_0 be a symbolic state

$$\begin{tikzcd}
 & & s_0 \arrow[dl] & \alpha_0 \arrow[dl, "c_1"] \\
 & s_1 \arrow[dl] & \alpha_1 \arrow[dl, "c_2"] & \\
s_2 \arrow[dr] & \alpha_2 \arrow[dr, "c_3"] & & \\
 & s_3 \arrow[dl] & \alpha_3 \arrow[dl, "c_4"] & \\
s_4 & \alpha_4 & &
\end{tikzcd}$$

– α_0 similar to s_0 but inputs unknown

– symbolic exec. mimicks the concrete one

– $c_1, \ldots, c_4$ constraints on the symb. values

– computing alternative (concrete) states:

$$\begin{array}{rcl}
\neg c_1 & \Rightarrow & s_0' \\
c_1 \wedge \neg c_2 & \Rightarrow & s_0'' \\
c_1 \wedge c_2 \wedge \neg c_3 & \Rightarrow & s_0''' \\
\ldots & &
\end{array}$$

Concolic testing: basic idea

Let s_0 be a concrete state Let α_0 be a symbolic state

$$\begin{tikzcd}
 & & s_0 \arrow[dl] & \alpha_0 \arrow[dl, "c_1"] \\
 & s_1 \arrow[dl] & \alpha_1 \arrow[dl, "c_2"] & \\
s_2 \arrow[dr] & \alpha_2 \arrow[dr, "c_3"] & & \\
 & s_3 \arrow[dl] & \alpha_3 \arrow[dl, "c_4"] & \\
s_4 & \alpha_4 & &
\end{tikzcd}$$

– α_0 similar to s_0 but inputs unknown

– symbolic exec. mimicks the concrete one

– $c_1, \ldots, c_4$ constraints on the symb. values

– computing alternative (concrete) states:

$$\begin{array}{rcl}
\neg c_1 & \Rightarrow & s_0' \\
c_1 \wedge \neg c_2 & \Rightarrow & s_0'' \\
c_1 \wedge c_2 \wedge \neg c_3 & \Rightarrow & s_0''' \\
\ldots & &
\end{array}$$

$$\begin{tikzcd}
 & main(2,3) \arrow[d] & & main(X,Y) \arrow[d] \\
 & if\ (X>0) \arrow[dl] & & if\ (X>0) \arrow[dl, "c_1 \equiv X > 0"] \\
if\ (Y<0) \arrow[dr] & & if\ (Y<0) \arrow[dr, "c_2 \equiv Y \geq 0"] & \\
 & skip & & skip
\end{tikzcd}$$

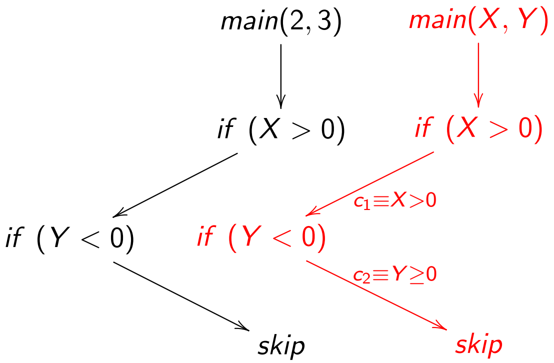
New test cases:

- $\neg c_1 \equiv \neg(X > 0) \equiv X \leq 0 \Rightarrow main(0,3)$
- $c_1 \wedge \neg c_2 \equiv (X > 0) \wedge \neg(Y \geq 0) \Rightarrow main(2,-1)$

$$\begin{tikzcd}
 & main(2,3) \arrow[d] & main(X,Y) \arrow[d] \\
 & if\ (X>0) \arrow[dl] & if\ (X>0) \arrow[dl, "c_1 \equiv X>0"] \\
if\ (Y<0) \arrow[dr] & if\ (Y<0) \arrow[dr, "c_2 \equiv Y \geq 0"] & \\
 & skip & skip
\end{tikzcd}$$

New test cases:

- $\neg c_1 \equiv \neg(X > 0) \equiv X \leq 0 \Rightarrow main(0, 3)$
- $c_1 \wedge \neg c_2 \equiv (X > 0) \wedge \neg(Y \geq 0) \Rightarrow main(2, -1)$



New test cases:

- $\neg c_1 \equiv \neg(X > 0) \equiv X \leq 0 \Rightarrow main(0, 3)$
- $c_1 \wedge \neg c_2 \equiv (X > 0) \wedge \neg(Y \geq 0) \Rightarrow main(2, -1)$

$$\begin{tikzcd}
 & main(2,3) \arrow[d] & & main(X,Y) \arrow[d] \\
 & if\ (X>0) \arrow[dl] & & if\ (X>0) \arrow[dl, "c_1 \equiv X>0"'] \\
if\ (Y<0) \arrow[dr] & & if\ (Y<0) \arrow[dr, "c_2 \equiv Y \geq 0"] & \\
 & skip & & skip
\end{tikzcd}$$

New test cases:

- $\neg c_1 \equiv \neg(X>0) \equiv X \leq 0 \Rightarrow main(0,3)$
- $c_1 \wedge \neg c_2 \equiv (X>0) \wedge \neg(Y \geq 0) \Rightarrow main(2,-1)$

Concolic testing in LP

The good news

- concrete execution = symbolic execution

Main differences

- unification, nondeterminism and backtracking
- the way input data to explore alternative paths is computed [ICLP 2015]

Concolic testing in LP

The good news

- concrete execution = symbolic execution

Main differences

- unification, nondeterminism and backtracking
- the way input data to explore alternative paths is computed [ICLP 2015]

Concolic execution in LP

(ℓ_1) $p(s(a)).$	(ℓ_4) $q(a).$	(ℓ_6) $r(a).$
(ℓ_2) $p(s(W)) \leftarrow q(W).$	(ℓ_5) $q(b).$	(ℓ_7) $r(c).$
(ℓ_3) $p(f(X)) \leftarrow r(X).$		

A concolic execution for, e.g., $p(f(a))$ will combine a concrete execution

$$p(f(a)) \rightarrow_{id} r(a) \rightarrow_{id} true$$

with a symbolic execution for $p(N)$:

$$p(N) \rightarrow_{\{N/f(Y)\}} r(Y) \rightarrow_{\{Y/a\}} true$$

that mimicks the steps of the former derivation despite being more general

The concolic execution actually looks like

$$\langle p(f(a))_{id} \;]\![\; p(N)_{id}\rangle \rightsquigarrow_{c(\{\ell_3\},\{\ell_1,\ell_2,\ell_3\})} \langle r(a)_{id} \;]\![\; r(Y)_{\{N/f(Y)\}}\rangle$$
$$\rightsquigarrow_{c(\{\ell_6\},\{\ell_6,\ell_7\})} \langle \mathrm{true}_{id} \;]\![\; \mathrm{true}_{\{N/f(a)\}}\rangle$$

Choice steps store the labels of the clauses that unified with each concrete and symbolic goals

Therefore, when looking for new run time goals that explore alternative paths, one should look for instances of $p(N)$ that unify with

- $\{\}$,
- $\{\ell_1\}$,
- $\{\ell_1, \ell_2\}$,
- $\{\ell_1, \ell_2, \ell_3\}$,
- $\{\ell_2\}$,
- ...

Selective Unification

Atom A	p(N)
Positive atoms $\mathcal{H}^+$	p(s(a))
Negative atoms $\mathcal{H}^-$	p(s(W)), p(f(X))

The concolic execution actually looks like

$$\langle p(f(a))_{id} \,[\![\, p(N)_{id} \rangle \leadsto_{c(\{\ell_3\},\{\ell_1,\ell_2,\ell_3\})} \langle r(a)_{id} \,[\![\, r(Y)_{\{N/f(Y)\}} \rangle$$
$$\leadsto_{c(\{\ell_6\},\{\ell_6,\ell_7\})} \langle \mathsf{true}_{id} \,[\![\, \mathsf{true}_{\{N/f(a)\}} \rangle$$

Choice steps store the labels of the clauses that unified with each concrete and symbolic goals

Therefore, when looking for new run time goals that explore alternative paths, one should look for instances of $p(N)$ that unify with

- $\{\}$,
- $\{\ell_1\}$,
- $\{\ell_1, \ell_2\}$,
- $\{\ell_1, \ell_2, \ell_3\}$,
- $\{\ell_2\}$,
- ...

Selective Unification

Atom A	p(N)
Positive atoms $\mathcal{H}^+$	p(s(a))
Negative atoms $\mathcal{H}^-$	p(s(W)), p(f(X))

The concolic execution actually looks like

$$\langle p(f(a))_{id} \rrbracket\, p(N)_{id}\rangle \rightsquigarrow_{c(\{\ell_3\},\{\ell_1,\ell_2,\ell_3\})} \langle r(a)_{id} \rrbracket\, r(Y)_{\{N/f(Y)\}}\rangle$$
$$\rightsquigarrow_{c(\{\ell_6\},\{\ell_6,\ell_7\})} \langle \mathsf{true}_{id} \rrbracket\, \mathsf{true}_{\{N/f(a)\}}\rangle$$

Choice steps store the labels of the clauses that unified with each concrete and symbolic goals

Therefore, when looking for new run time goals that explore alternative paths, one should look for instances of $p(N)$ that unify with

- $\{\}$,
- $\{\ell_1\}$,
- $\{\ell_1, \ell_2\}$,
- $\{\ell_1, \ell_2, \ell_3\}$,
- $\{\ell_2\}$,
- ...

$\boxed{\text{Selective Unification}}$

Atom A	p(N)
Positive atoms $\mathcal{H}^+$	p(s(a))
Negative atoms $\mathcal{H}^-$	p(s(W)), p(f(X))

The concolic execution actually looks like

$$\langle p(f(a))_{id} \,[\![\, p(N)_{id} \rangle \leadsto_{c(\{\ell_3\},\{\ell_1,\ell_2,\ell_3\})} \langle r(a)_{id} \,[\![\, r(Y)_{\{N/f(Y)\}} \rangle$$
$$\leadsto_{c(\{\ell_6\},\{\ell_6,\ell_7\})} \langle \mathsf{true}_{id} \,[\![\, \mathsf{true}_{\{N/f(a)\}} \rangle$$

Choice steps store the labels of the clauses that unified with each concrete and symbolic goals

Therefore, when looking for new run time goals that explore alternative paths, one should look for instances of $p(N)$ that unify with

- $\{\}$,
- $\{\ell_1\}$,
- $\{\ell_1, \ell_2\}$,
- $\{\ell_1, \ell_2, \ell_3\}$,
- $\{\ell_2\}$,
- ...

$\boxed{\text{Selective Unification}}$

Atom A	p(N)
Positive atoms $\mathcal{H}^+$	p(s(a))
Negative atoms $\mathcal{H}^-$	p(s(W)), p(f(X))

The Selective Unification Problem – ICLP15

- A be an atom
- $G \subseteq \mathcal{V}ar(A)$ be a set of variables (when ground, the initial goal terminates)
- $\mathcal{H}^+$ and $\mathcal{H}^-$ be finite sets of atoms such that all atoms are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+ \cup \mathcal{H}^-$

"$A \approx B$" stands for "A unifies with B"

Definition (selective unification problem)

$$\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) = \left\{ \sigma\restriction_{\mathcal{V}ar(A)} \;\middle|\; \begin{array}{l} \quad \forall H \in \mathcal{H}^+ : A\sigma \approx H \\ \wedge\ \forall H \in \mathcal{H}^- : \neg(A\sigma \approx H) \\ \wedge\ G\sigma \text{ is ground} \end{array} \right\}$$

The set $\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$ can be infinite

Examples

$$\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) = \left\{ \sigma\restriction_{\mathcal{V}ar(A)} \;\middle|\; \begin{array}{l} \quad \forall H \in \mathcal{H}^+ : A\sigma \approx H \\ \wedge\ \forall H \in \mathcal{H}^- : \neg(A\sigma \approx H) \\ \wedge\ G\sigma \text{ is ground} \end{array} \right\}$$

- $A = p(X)$, $\mathcal{H}^+ = \{p(a), p(b)\}$, $\mathcal{H}^- = \emptyset$, $G = \emptyset$
  One solution: ϵ, $p(X)$ unifies with $p(a)$ and $p(b)$
- $A = p(X)$, $\mathcal{H}^+ = \{p(a), p(b)\}$, $\mathcal{H}^- = \{p(f(Z))\}$, $G = \emptyset$
  No solution: there is no instance of A that unifies with both atoms in $\mathcal{H}^+$ and does not unify with $p(f(Z))$
- $A = p(X)$, $\mathcal{H}^+ = \{p(s(Y))\}$, $\mathcal{H}^- = \{p(s(0))\}$, $G = \{X\}$
  Infinitely many solutions, including $\{X/s^{n+2}(0)\}$ for $n \in \mathbb{N}$
  E.g., $\sigma = \{X/s(s(0))\}$, $A\sigma = p(s(s(0)))$, $A\sigma$ and $p(s(Y))$ unify, $A\sigma$ and $p(s(0))$ do not, $X\sigma$ is ground
- $A = p(X, Y)$, $\mathcal{H}^+ = \{p(a, b), p(Z, Z)\}$, and $\mathcal{H}^- = \emptyset$, $G = \emptyset$
  Two solutions: $\{X/a\}$ and $\{Y/b\}$

Finite signatures – LOPSTR16

Theorem

For finite signatures, $\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) \neq \emptyset$ is decidable

Idea: when the signature is finite

- there exists n such that, if a solution has not been found when considering terms of depth $\leq n$, then the problem is not satisfiable
- hence a bounded generate-and-test algorithm is sound and complete

Infinite signatures – LOPSTR16

- Linearity = each variable occurs only once
- We restrict our interest to linear solutions

Definition (selective linear unification problem – SLUP)

$$\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) = \left\{ \sigma\restriction_{\mathcal{V}ar(A)} \;\middle|\; \begin{array}{l} \quad \forall H \in \mathcal{H}^+ : A\sigma \approx H \\ \wedge\ \forall H \in \mathcal{H}^- : \neg(A\sigma \approx H) \\ \wedge\ G\sigma \text{ is ground} \\ \wedge\ \sigma \text{ is linear} \end{array} \right\}$$

- We only consider linear sets of positive atoms $\mathcal{H}^+$
- We present a sound and complete algorithm for SLUP

Definitions

- A structure $\mathcal{D}$ admits *quantifier elimination* if for each first-order formula φ there exists a quantifier-free formula ψ such that $\mathcal{D} \models \forall[\varphi \leftrightarrow \psi]$
- A *constraint atom* is a tuple of the form $\langle c \mid p(\vec{X})\rangle$ where $\vec{X}$ is a vector of distinct variables and c is a constraint

CSUP

Let

- A be a constraint atom of the form $\langle c_A \,|\, p(\vec{X})\rangle$ with $G \subseteq \mathcal{V}ar(A)$
- $\mathcal{H}^+$ and $\mathcal{H}^-$ be finite sets of constraint atoms such that all constraint atoms, including A, are pairwise variable disjoint and $A \approx B$ for all $B \in \mathcal{H}^+ \cup \mathcal{H}^-$

Definition (constraint selective unification problem – CSUP)

$\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) =$

$$\left\{ c_A \wedge \mathbf{c} \;\middle|\; \begin{array}{l} \quad c_A \wedge c \text{ is satisfiable} \\ \wedge\ c \text{ is variable disjoint with } \mathcal{H}^+ \cup \mathcal{H}^- \\ \wedge\ \forall H \in \mathcal{H}^+ : \langle c_A \wedge c \,|\, p(\vec{X})\rangle \approx H \\ \wedge\ \forall H \in \mathcal{H}^- : \neg(\langle c_A \wedge c \,|\, p(\vec{X})\rangle \approx H) \\ \wedge\ \text{each } X \in G \text{ is fixed within } c_A \wedge c \end{array} \right\}$$

Some CSUP for CLP($\mathcal{Q}_{lin}$)

$$\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) = \left\{ c_A \wedge \mathbf{c} \;\middle|\; \begin{array}{l} \quad c_A \wedge c \text{ is satisfiable} \\ \wedge\ c \text{ is variable disjoint with } \mathcal{H}^+ \cup \mathcal{H}^- \\ \wedge\ \forall H \in \mathcal{H}^+ : \langle c_A \wedge c \,|\, p(\vec{X})\rangle \approx H \\ \wedge\ \forall H \in \mathcal{H}^- : \neg(\langle c_A \wedge c \,|\, p(\vec{X})\rangle \approx H) \\ \wedge\ \text{each } X \in G \text{ is fixed within } c_A \wedge c \end{array} \right\}$$

- $A = \langle 0 \leq X \wedge X \leq 5 \,|\, p(X)\rangle$, $\mathcal{H}^+ = \{\langle 4 \leq Y \,|\, p(Y)\rangle\}$, $\mathcal{H}^- = \{\langle Z < 2 \,|\, p(Z)\rangle\}$, $G = \{X\}$
  Infinitely many solutions, e.g., $c = (X = 9/2)$
  $0 \leq X \wedge X \leq 5 \wedge X = 9/2$ sat, $X = Y, 4 \leq Y, X = 9/2$ sat, $X = Z, Z < 2, X = 9/2$ unsat and X is ground

- $A = \langle 0 \leq X \wedge X \leq 5 \,|\, p(X)\rangle$, $\mathcal{H}^+ = \{\langle 4 \leq Y_1 \,|\, p(Y_1)\rangle, \langle Y_2 \leq 1 \,|\, p(Y_2)\rangle\}$, $\mathcal{H}^- = \{\langle 2 < Z \wedge Z < 3 \,|\, p(Z)\rangle\}$, $G = \emptyset$
  No solution

Theorem

For CLP in general, $\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G) \neq \emptyset$ *is undecidable*

Idea: encode the halting problem for Turing machines in CLP($\mathcal{A}$), where $\mathcal{A}$ is a subclass of the decidable *array property* fragment introduced in [BradleyMS06]

Additional hypotheses

A1: The constraint structure admits variable elimination

A2: The negation of any atomic constraint is equivalent to a finite disjunction of atomic constraints

Example

$\mathcal{Q}_{lin}$ with $\{< /2, \leq /2, = /2, \geq /2, > /2\}$ verifies A1 and A2:

- Fourier-Motzkin for variable elimination
- The negation of each atomic constraint from $\{< /2, \leq /2, \geq /2, > /2\}$ is an atomic constraint
- $\neg(X = Y) \equiv X < Y \vee X > Y$

CSUP without the Groundness Condition

Algorithm CSUP-$(A, \mathcal{H}^+, \mathcal{H}^-)$ terminates, correct and complete

1. Intersect of all the complements of the atoms in $\mathcal{H}^-$:

$$I := \bigwedge\{\neg C' | H = \langle c' | p(\vec{Y})\rangle \in \mathcal{H}^-, C' \equiv \exists \vec{Y}[\vec{X} = \vec{Y} \wedge c']\}$$

2. Eliminate negation from I then distribute $\wedge$ over $\vee$: $J := \bigvee_{1 \leq j \leq n} C_j(\vec{X})$
3. Intersect J with A: $K := \bigvee_{1 \leq j \leq n}[C_j(\vec{X}) \wedge c_A]$
4. Collect the constraints from K which intersect each of $\mathcal{H}^+$:

$$S := \left\{ C_j(\vec{X}) \wedge c_A \in K \;\middle|\; \bigwedge_{\langle c' | p(\vec{X}')\rangle \in \mathcal{H}^+} \mathcal{D} \models \exists[\vec{X}' = \vec{X} \wedge C_j^A(\vec{X}) \wedge c'] \right\}$$

5. Return S

Step 2 relies on A1 and A2

CSUP with the Groundness Condition

Algorithm CSUP terminates, correct and complete

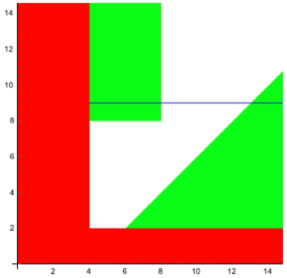
Postcondition: A possibly empty finite set of constraints, each of them being a solution of $\mathcal{P}(A, \mathcal{H}^+, \mathcal{H}^-, G)$

```
1 $S := CSUP^-(A, \mathcal{H}^+, \mathcal{H}^-)$
2 $T := \emptyset$
3 For each $C_j \in S$ do
    1 $U := GRND(\langle C_j | p(\vec{X}) \rangle, \mathcal{H}^+, G)$
    2 If $U \neq \perp$ then $T := T \cup \{C_j \wedge U\}$
4 Return $T$
```

The function GRND is domain dependent

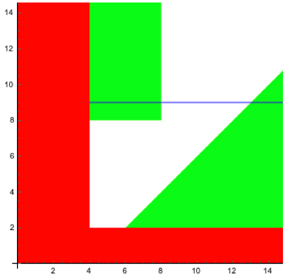
An example

- $A := \langle c_A \,|\, p(X, Y)\rangle$, with $c_A \equiv 0 \leq X \wedge 0 \leq Y$
- $\mathcal{H}^+ := \left\{ \begin{array}{l} \langle Y1 \leq X1 - 4 \,|\, p(X1, Y1)\rangle, \\ \langle X2 \leq 8 \wedge 8 \leq Y2 \,|\, p(X2, Y2)\rangle \end{array} \right\}$,
- $\mathcal{H}^- := \{\langle Y3 \leq 2 \,|\, p(X3, Y3)\rangle, \langle X4 \leq 4 \,|\, p(X4, Y4)\rangle\}$.



Geometrical interpretation:

- the first quadrant of the plane (restricted to $X < 15$ and $Y < 15$) as the solutions are inside c_A
- the two positive spaces:
  - $Y \leq X - 4$ in the lower right
  - $X \leq 8 \wedge 8 \leq Y$ in the upper left
- the two negative spaces:
  - $Y \leq 2$
  - $X \leq 4$



- For $G = \emptyset$, we get $c_A \wedge \{4 < X \wedge 2 < Y\}$
  The union of the two green areas with the white one in between
  It has a non-empty intersection with the positive spaces and an empty intersection with the negative spaces
- For $G = \{Y\}$, we get $c_A \wedge \{4 < X \wedge Y = 9\}$
  The blue half-line
  The half-line – included into the first quadrant and with Y ground – has a non-empty intersection with both positive spaces and an empty intersection with the negative spaces
- For $G = \{X\}$, $c_A \wedge \{X = 7 \wedge 2 < Y\}$ Idem
- For $G = \{X, Y\}$, $\emptyset$
  Can one find a point which belongs to the green upper left space and at the same time to the green lower right space? No

Implemented

The queries solving the previous example:

```
?- csup(p(X,Y)-[X>=0,Y>=0],
            [p(X1,Y1)-[Y1=<X1-4], p(X2,Y2)-[X2=<8,Y2>=8]],
            [p(X3,Y3)-[Y3=<2],p(X4,Y4)-[X4=<4]],
            [], S).
S = p(X, Y)-[Y>2, X>4].

?- csup(p(X,Y)-[X>=0,Y>=0],
            [p(X1,Y1)-[Y1=<X1-4], p(X2,Y2)-[X2=<8,Y2>=8]],
            [p(X3,Y3)-[Y3=<2],p(X4,Y4)-[X4=<4]],
            [Y],S).
S = p(X, Y)-[Y=9, X>4].

?- csup(p(X,Y)-[X>=0,Y>=0],
            [p(X1,Y1)-[Y1=<X1-4], p(X2,Y2)-[X2=<8,Y2>=8]],
            [p(X3,Y3)-[Y3=<2],p(X4,Y4)-[X4=<4]],
            [X],S).
S = p(X, Y)-[X=7, Y>2].

?- csup(p(X,Y)-[X>=0,Y>=0],
            [p(X1,Y1)-[Y1=<X1-4], p(X2,Y2)-[X2=<8,Y2>=8]],
            [p(X3,Y3)-[Y3=<2],p(X4,Y4)-[X4=<4]],
            [X,Y],S).
false.
```

Summary

- We have considered concolic testing for CLP
- We have proved that the selective unification problem is generally undecidable for CLP
- For a restricted class of constraint structures, we have given a generic correct and complete algorithm for selective unification without the groundness condition
- For CLP($\mathcal{Q}_{lin}$), we have presented a specific correct and complete selective unification with the groundness condition
- Future work: investigate the links with constructive negation

Thank you for your attention!