Mixing Polyedra and Boxes Abstract Domain for Constraint Solving

Marie Pelleau$^{1,2}$ Emmanuel Rauzy$^1$ Ghiles Ziat$^2$
Charlotte Truchet$^3$ Antoine Miné$^2$

1. École Normale Supérieure, France
2. Université Pierre et Marie Curie, France
3. IRISA, INRIA Rennes Bretagne Atlantique, France

CP meets Verification 2016
September 5, 2016
Outline

1. Context
   - Constraint Programming
   - Abstract Interpretation
   - Comparison

2. Abstract Solving Method

3. AbSolute
Constraint Programming (CP) formalizes and solves combinatorial problems [Montanari, 1974]

- Declarative programming, specify the problem not the solving method
- Use to solve many industrial problems
  - In biology (e.g. ARN secondary structure [Perriquet and Barahona, 2009])
  - In logistics (e.g. job shop scheduling problem [Grimes and Hebrard, 2011])
  - In verification (e.g. program verification [Collavizza and Rueher, 2007], model verification [Lazaar et al., 2012])
  - In test generation (e.g. automatic generation of pairwise configuration tests [Hervieu et al., 2011])
  - In cryptography (e.g. design of cryptographic s-boxes [Ramamoorthy et al., 2011])
  - In music [Truchet and Assayag, 2011]
Constraint Satisfaction Problem (CSP)

Definition (CSP)
- \( V \): set of variables
- \( D \): set of domains
- \( C \): set of constraints

Example (Continuous)
- \( V = (v_1, v_2) \)
- \( D_1 = [-1, 14], D_2 = [-5, 10] \)
- \( C_1 : (v_1 - 9)^2 + v_2^2 \leq 25 \)
- \( C_2 : (v_1 + 1)^2 + (v_2 - 5)^2 \leq 100 \)
Definition (Exact Solution)

An exact solution is an instantiation of the variables satisfying all the constraints.

Remark

Computing the exact solutions can be too expensive or intractable.

Definition (Approximated Solution)

The solution set is approximated by a set of boxes that only contain solutions or are small enough w.r.t. a parameter $r$. 
Solving Method

How to solve this?

Propagation
Using the constraints, deletes from the domains the values that cannot be part of a solution

Exploration
Splits a box into two smaller boxes
Continuous Solving Method

Parameter: float r

list of boxes sols ← ∅
queue of boxes toExplore ← ∅
box e

e ← D
push e in toExplore

while toExplore ≠ ∅ do
    e ← pop(toExplore)
    e ← Hull-Consistency(e)
    if e ≠ ∅ then
        if maxDim(e) ≤ r or isSol(e) then
            sols ← sols ∪ e
        else
            split e in two boxes e1 and e2
            push e1 and e2 in toExplore

M. Pelleau, E. Rauzy, G. Ziat, C. Truchet, A. Miné
September 5, 2016
Continuous Solving Method

Parameter: float r

list of boxes sols ← ∅
queue of boxes toExplore ← ∅
box e

e ← D
push e in toExplore

while toExplore ≠ ∅ do
    e ← pop(toExplore)
    e ← Hull-Consistency(e)
    if e ≠ ∅ then
        if maxDim(e) ≤ r or isSol(e) then
            sols ← sols ∪ e
        else
            split e in two boxes e1 and e2
            push e1 and e2 in toExplore
    else

Continuous Solving Method

Parameter: \( \text{float } r \)

list of boxes \( \text{sols} \) ← ∅
queue of boxes \( \text{toExplore} \) ← ∅
box \( e \)

\( e \) ← \( D \)
\text{push } e \text{ in } \text{toExplore}

\text{while } \text{toExplore} \neq \emptyset \text{ do}
\( e \) ← \text{pop}(\text{toExplore})
\( e \) ← \text{Hull-Consistency}(e)
\text{if } e \neq \emptyset \text{ then}
\text{if } \text{maxDim}(e) \leq r \text{ or } \text{isSol}(e) \text{ then}
\( \text{sols} \) ← \( \text{sols} \cup e \)
\text{else}
\text{split } e \text{ in two boxes } e_1 \text{ and } e_2
\text{push } e_1 \text{ and } e_2 \text{ in } \text{toExplore}

Continuous Solving Method

Parameter: float \( r \)

list of boxes \( \text{sols} \) ← \( \emptyset \)
queue of boxes \( \text{toExplore} \) ← \( \emptyset \)
box \( e \)

\( e \) ← \( D \)
push \( e \) in toExplore

while \( \text{toExplore} \neq \emptyset \) do
  \( e \) ← pop(\( \text{toExplore} \))
  \( e \) ← Hull-Consistency(\( e \))
  if \( e \neq \emptyset \) then
    if \( \text{maxDim}(e) \leq r \) or isSol(\( e \)) then
      \( \text{sols} \) ← \( \text{sols} \cup e \)
    else
      split \( e \) in two boxes \( e_1 \) and \( e_2 \)
push \( e_1 \) and \( e_2 \) in toExplore
Continuous Solving Method

Parameter: float r

list of boxes sols ← ∅
queue of boxes toExplore ← ∅
box e

e ← D
push e in toExplore

while toExplore ≠ ∅ do
    e ← pop(toExplore)
e ← Hull-Consistency(e)
    if e ≠ ∅ then
        if maxDim(e) ≤ r or isSol(e) then
            sols ← sols ∪ e
        else
            split e in two boxes e1 and e2
            push e1 and e2 in toExplore
Continuous Solving Method

Parameter: float r

list of boxes sols ← ∅
queue of boxes toExplore ← ∅
box e

e ← D
push e in toExplore

while toExplore ≠ ∅ do
  e ← pop(toExplore)
e ← Hull-Consistency(e)
  if e ≠ ∅ then
    if maxDim(e) ≤ r or isSol(e) then
      sols ← sols ∪ e
    else
      split e in two boxes e1 and e2
      push e1 and e2 in toExplore
  else

---

M. Pelleau, E. Rauzy, G. Ziat, C. Truchet, A. Miné
September 5, 2016 6 / 23
Synthesis

What CP does

- Offers a framework to model many combinatorial problems
- Solves problems on either discrete or continuous domains
- Has various heuristics to improve the solving methods

⇒ Efficiently solves many combinatorial problems

What CP does not

- Take into account the correlation of the variables ⇒ restricted to Cartesian product
- Solve mixed discrete-continuous problems

Remark

Computes over-approximations of the solution set
Abstract Interpretation

Remark

Other domain that computes over-approximations

- Abstract Interpretation (AI) is a theory of approximation of the semantics [Cousot and Cousot, 1976]
- Applied to the static analysis and verification of softwares
- Main application: automatically prove that a program does not have execution errors
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y<3 and x≤8 do
  5: x ← x+y
  6: y ← 2*y
  7: x ← x-1
  8: y ← y+1
Abstract Interpretation

Study the variables values

```plaintext
1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y < 3 and x ≤ 8 do
5:   x ← x + y
6:   y ← 2*y
7:   x ← x - 1
8:   y ← y + 1
```

Forbidden Zone

Concrete domain $D^\blacklozenge$

Abstract domain $D^\blackloshoe$

False Alarm

Remark

Approximation with various abstract domains

Tradeoff between expressivity and cost of an abstract domain
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y < 3 and x ≤ 8 do
5:   x ← x+y
6:   y ← 2*y
7:   x ← x-1
8: y ← y+1

Remark
Computing in the concrete domain can be undecidable or too expensive
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y<3 and x≤8 do
5:   x ← x+y
6:   y ← 2*y
7:   x ← x-1
8:   y ← y+1

Abstract domain $\mathcal{D}^\#$
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y < 3 and x ≤ 8 do
5:   x ← x + y
6:   y ← 2*y
7: x ← x - 1
8: y ← y + 1
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y<3 and x≤8 do
5:   x ← x+y
6:   y ← 2*y
7:   x ← x-1
8:   y ← y+1

False Alarm
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y < 3 and x ≤ 8 do
5: x ← x + y
6: y ← 2*y
7: x ← x - 1
8: y ← y + 1
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y<3 and x≤8 do
5:   x ← x+y
6:   y ← 2*y
7:  x ← x-1
8:  y ← y+1
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y < 3 and x ≤ 8 do
  5: x ← x + y
  6: y ← 2 * y
  7: x ← x - 1
  8: y ← y + 1
Abstract Interpretation

Study the variables values

1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y<3 and x≤8 do
5:   x ← x+y
6:   y ← 2*y
7:   x ← x-1
8:   y ← y+1

Remark
- Approximation with various abstract domains
- Tradeoff between expressivity and cost of an abstract domain
Comparison

- Same underlying structure (lattices and fixpoints)
- Same goal: an over-approximation of a desired set
  - Solutions set in CP
  - Environments set in AI

- Different fixpoints
  - Greatest fixpoint in CP
  - Least and greatest fixpoint in AI

- Different iterative schemes
  - Only decreasing iterations in CP
  - Both decreasing and increasing iterations in AI

- No precision function in AI
- More domains representations in AI than in CP
- AI deals naturally with mixed discrete-continuous domains
Bringing together AI and CP

- Improvement of static analyser [Ponsini et al., 2011]
- Feature models analysis and automatic generation of configuration tests [Hervieu et al., 2011]
- Galois connection in CP [Scott, 2016]

Previous work
- Define abstract domains in CP [Pelleau et al., 2014]
- Use abstract domains in a solving method [Pelleau et al., 2011]
- Define a solving method using AI tools [Pelleau et al., 2013]

Our contribution
- Use reduced product
- Visualization tool
Introduced in [Cousot and Cousot, 1979]

An abstract domain can be a product of abstract domains

Reduced product propagates information from one domain to another
Continuous Solving Method

Parameter: float $r$

list of boxes $sols \leftarrow \emptyset$
queue of boxes $toExplore \leftarrow \emptyset$
box $e \leftarrow D$

push $e$ in $toExplore$

while $toExplore \neq \emptyset$ do
    $e \leftarrow \text{pop}(toExplore)$
    $e \leftarrow \text{Hull-Consistency}(e)$
    if $e \neq \emptyset$ then
        if $\text{maxDim}(e) \leq r$ or $\text{isSol}(e)$ then
            $sols \leftarrow sols \cup e$
        else
            split $e$ in two boxes $e_1$ and $e_2$
            push $e_1$ and $e_2$ in $toExplore$

Under some conditions on the operators, this abstract solving method terminates, is correct and complete.
Abstract Solving Method

Parameter: float $r$

list of boxes disjunction sols $\leftarrow \emptyset$
queue of boxes disjunction toExplore $\leftarrow \emptyset$
box abstract domain $e \leftarrow \mathcal{D} \top$

push $e$ in toExplore

while toExplore $\neq \emptyset$ do
  $e \leftarrow \text{pop}(\text{toExplore})$
  $e \leftarrow \text{Hull-Consistency}(e)$  $\rho^\#(e)$
  if $e \neq \emptyset$ then
    if maxDim($e$) $\tau(e)$ $\leq r$ or isSol($e$) then
      sols $\leftarrow$ sols $\cup$ $e$
    else
      split $e$ in two boxes $e_1$ and $e_2$
      push $e_1$ and $e_2$ $\oplus(e)$ in toExplore

Under some conditions on the operators, this abstract solving method terminates, is correct and complete.
AbSolute

Solver based on Apron [Jeannet and Miné, 2009], an OCaml library of numerical abstract domains for static analysis

- Consistency: using transfer functions
- Propagation loop: at each iteration, propagate all the constraints
  → Apply all the transfer functions

https://github.com/mpelleau/AbSolute.git
Experiments

- Problems from the COCONUT benchmark
- Comparison with Ibex [Chabert and Jaulin, 2009]
- Same configuration
## Results

<table>
<thead>
<tr>
<th>problem</th>
<th>#var</th>
<th>#ctrs</th>
<th>type</th>
<th>AbS</th>
<th>Ibex</th>
</tr>
</thead>
<tbody>
<tr>
<td>bronstein</td>
<td>3</td>
<td>3</td>
<td>=</td>
<td>16.905</td>
<td>13.990</td>
</tr>
<tr>
<td>brent8</td>
<td>8</td>
<td>8</td>
<td>=</td>
<td>3945.026</td>
<td>181.412</td>
</tr>
<tr>
<td>bellido</td>
<td>9</td>
<td>9</td>
<td>=</td>
<td>36.873</td>
<td>223.380</td>
</tr>
<tr>
<td>st_miqp5</td>
<td>7</td>
<td>15</td>
<td>≤</td>
<td>174.661</td>
<td>2135.269</td>
</tr>
<tr>
<td>hs5</td>
<td>2</td>
<td>1</td>
<td>≤</td>
<td>81.672192</td>
<td>152.101994</td>
</tr>
<tr>
<td>booth</td>
<td>2</td>
<td>1</td>
<td>≤</td>
<td>1069.671</td>
<td>1608.732</td>
</tr>
<tr>
<td>supersim</td>
<td>2</td>
<td>3</td>
<td>=, ≤</td>
<td>7.116</td>
<td>8.472</td>
</tr>
<tr>
<td>aljazzaf</td>
<td>3</td>
<td>2</td>
<td>=, ≤</td>
<td>4.714</td>
<td>18.057</td>
</tr>
</tbody>
</table>

CPU time in seconds to find all the solutions
Experiments

- Problems from the MinLPLib benchmark
- Same configuration
## Results

<table>
<thead>
<tr>
<th>problem</th>
<th>#var</th>
<th>#ctrs</th>
<th>type</th>
<th>boxes</th>
<th>prod</th>
</tr>
</thead>
<tbody>
<tr>
<td>mickey</td>
<td>2</td>
<td>2</td>
<td>~</td>
<td>49.015999</td>
<td>111.618042</td>
</tr>
<tr>
<td>eqlin</td>
<td>3</td>
<td>3</td>
<td>/</td>
<td>130.645037</td>
<td>4.138947</td>
</tr>
<tr>
<td>hs-f1</td>
<td>2</td>
<td>2</td>
<td>/, ~</td>
<td>4.773140</td>
<td>2.788067</td>
</tr>
<tr>
<td>octo_hole</td>
<td>2</td>
<td>8</td>
<td>/, ~</td>
<td>858.849049</td>
<td>510.193110</td>
</tr>
</tbody>
</table>

CPU time in seconds to find all the solutions
In 2D, problems with only two variables or projection on two variables
Visualization

In 3D, problems with three variables
Conclusion

- CP solving method can be defined with AI tools and techniques
- Abstract solving method is modular
- Hybrid CP–AI solver naturally handles mixed constraint problems
- Need to implement advanced CP heuristics in AbSolute
Perspectives

- Improve AbSolute using CP heuristics and techniques
  - specialized propagators
  - propagation loop
- Develop abstract domains for specific constraint
- Use CP methods in a AI-based static analyser
  - decreasing iteration methods (alternative to narrowing)
  - split operator in disjunctive completion
  - refine an abstract element to achieve completeness
- Use the widening in CP solver
Thank you for your attention!

Do you have questions?


Refining abstract interpretation-based approximations with constraint solvers.
In Proceedings of the 4th International Workshop on Numerical Software Verification.

The design of cryptographic s-boxes using csps.

Other Things Besides Number: Abstraction, Constraint Propagation, and String Variables.
PhD thesis, University of Uppsala.

Constraint Programming in Music.

ISTE.