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

- 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]

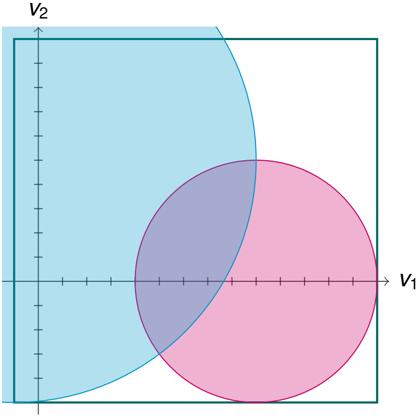
Constraint Satisfaction Problem (CSP)

Definition (CSP)

- V: set of variables
- D: set of domains
- C: set of constraints

Example (Continuous)

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



Constraint Satisfaction Problem (CSP)

Definition (Exact Solution)

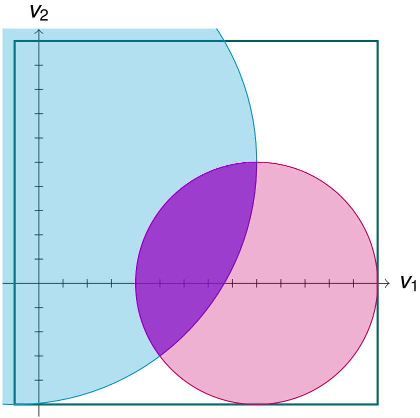
An exact solution is an instantiation of the variables satisfying all the constraints

Remark

Computing the exact solutions can be too expensive or intractable

Definition (Approximated Solution)

The solution set is approximated by an 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

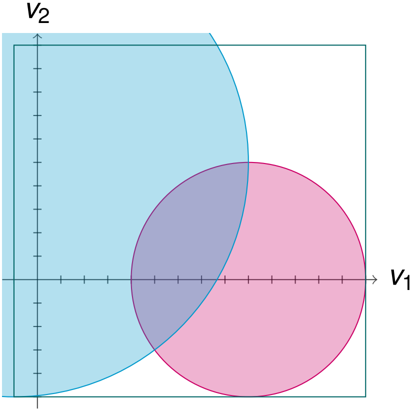
Continuous Solving Method

```
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
```



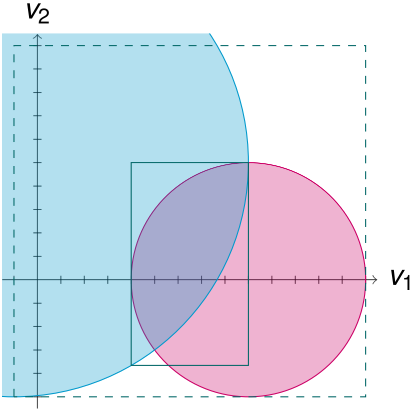
Continuous Solving Method

```
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
```



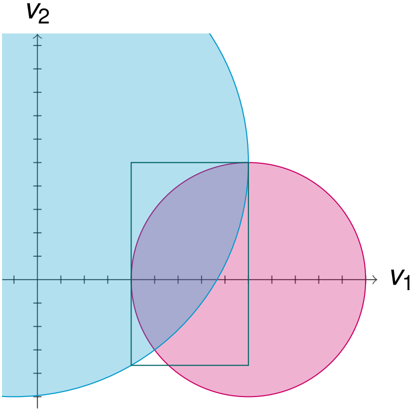
Continuous Solving Method

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
```



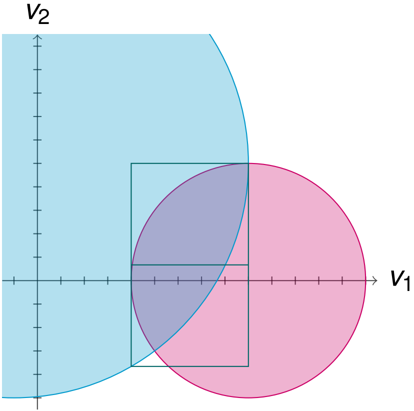
Continuous Solving Method

```
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
```



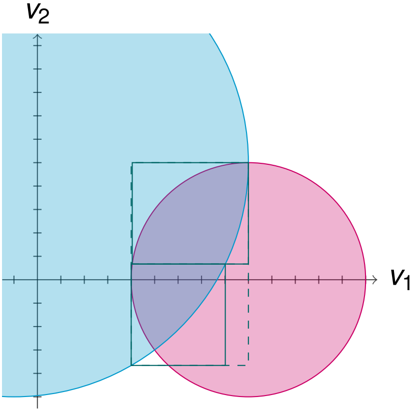
Continuous Solving Method

```
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
```



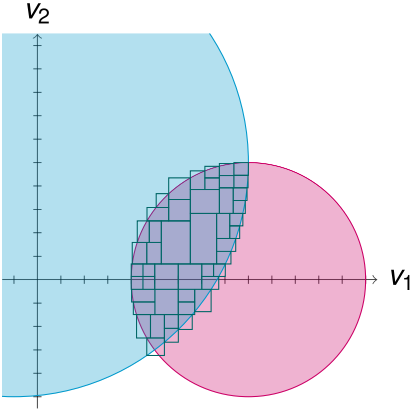
Continuous Solving Method

```
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
```



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

$\Longrightarrow$ Efficiently solves many combinatorial problems

What CP does not

- Take into account the correlation of the variables $\Rightarrow$ 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

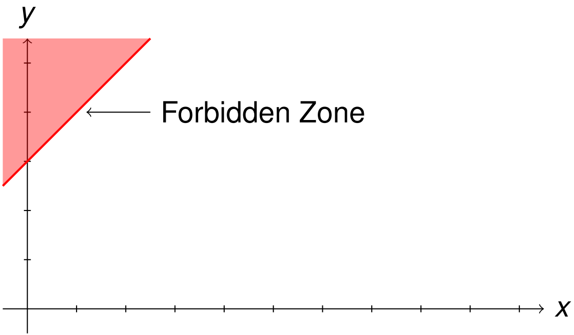
Study the variables values

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

Abstract Interpretation

Study the variables values

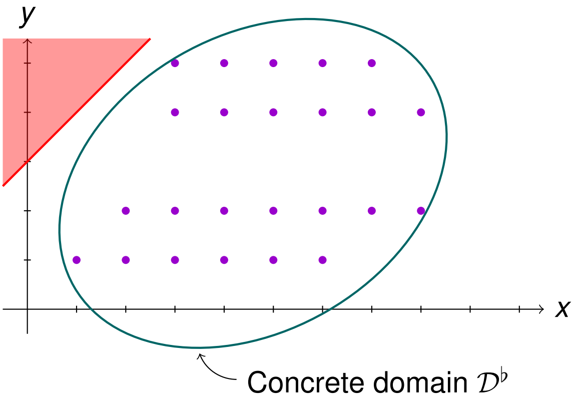
```
int x, y
y ← 1
x ← random(1, 5)
while y<3 and x≤8 do
  x ← x+y
  y ← 2*y
x ← x-1
y ← y+1
```



Abstract Interpretation

Study the variables values

```
int x, y
y ← 1
x ← random(1, 5)
while y<3 and x≤8 do
  x ← x+y
  y ← 2*y
x ← x−1
y ← y+1
```



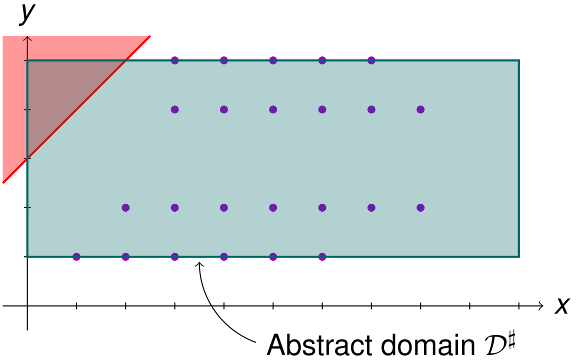
Remark

Computing in the concrete domain can be undecidable or too expensive

Abstract Interpretation

Study the variables values

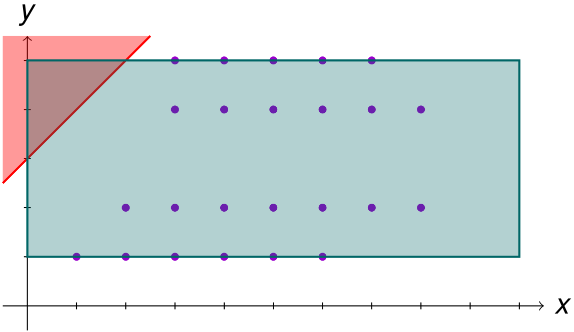
```
1: int x, y
2: y ← 1
3: x ← random(1, 5)
4: while y<3 and x≤8 do
5:    x ← x+y
6:    y ← 2*y
7: x ← x-1
8: y ← y+1
```



Abstract Interpretation

Study the variables values

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

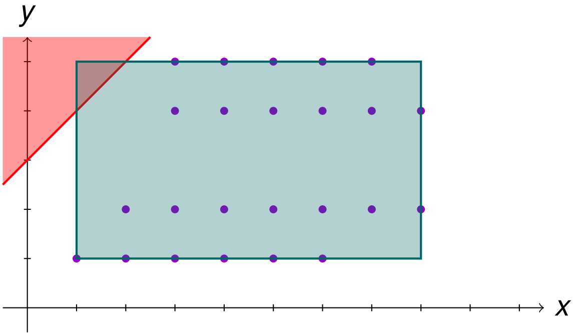


False Alarm

Abstract Interpretation

Study the variables values

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

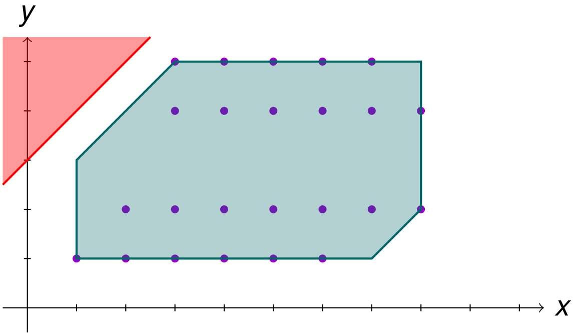


False Alarm

Abstract Interpretation

Study the variables values

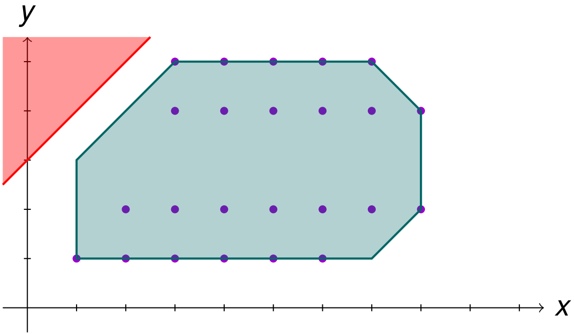
```
int x, y
y ← 1
x ← random(1, 5)
while y<3 and x≤8 do
  x ← x+y
  y ← 2*y
x ← x-1
y ← y+1
```



Abstract Interpretation

Study the variables values

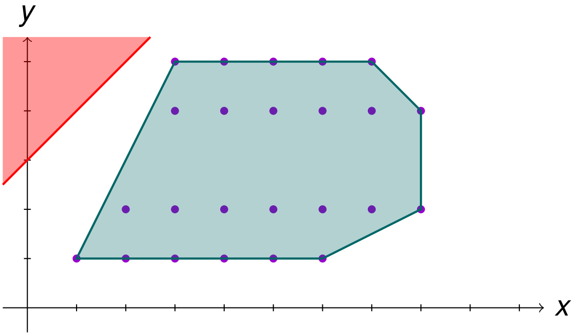
```
int x, y
y ← 1
x ← random(1, 5)
while y<3 and x≤8 do
   x ← x+y
   y ← 2*y
x ← x-1
y ← y+1
```



Abstract Interpretation

Study the variables values

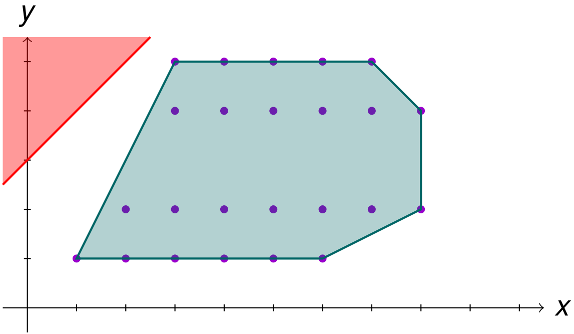
```
int x, y
y ← 1
x ← random(1, 5)
while y<3 and x≤8 do
    x ← x+y
    y ← 2*y
x ← x-1
y ← y+1
```



Abstract Interpretation

Study the variables values

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

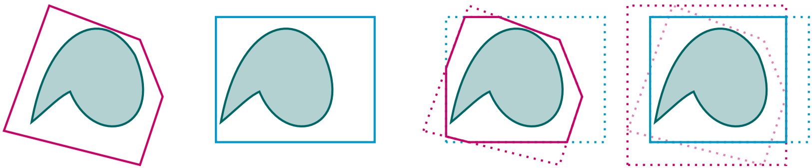
- Define abstract domains in CP [Pelleau et al., 2014]
- Use abstract domains in a solving method [Pelleau et al., 2011]
- Define a solving method using AI tools [Pelleau et al., 2013]

Our contribution

- Use reduced product
- Visualization tool

Mixing Abstract Domains

- 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



Polyhedron   Box   Reduced Product

Continuous Solving Method

```
Parameter:  float r

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

Abstract Solving Method

```
Parameter:  float r

~~list of boxes~~ disjunction sols ← ∅
~~queue of boxes~~ disjunction toExplore ← ∅
~~box~~ abstract domain e ← ~~D~~ ⊤♯

push e in toExplore

while toExplore ≠ ∅ do
  e ← pop(toExplore)
  e ← ~~Hull-Consistency(e)~~ ρ♯(e)
  if e ≠ ∅ then
    if ~~maxDim(e)~~ τ(e) ≤ r or isSol(e) then
      sols ← sols ∪ e
    else
      ~~split e in two boxes e1 and e2~~
      push ~~e1 and e2~~ ⊕(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
  $\longrightarrow$ 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

problem	#var	#ctrs	type	AbS	Ibex
bronstein	3	3	$=$	16.905	**13.990**
brent8	8	8	$=$	3945.026	**181.412**
bellido	9	9	$=$	**36.873**	223.380
st_miqp5	7	15	$\leq$	**174.661**	2135.269
hs5	2	1	$\leq$	**81.672192**	152.101994
booth	2	1	$\leq$	**1069.671**	1608.732
supersim	2	3	$=, \leq$	**7.116**	8.472
aljazzaf	3	2	$=, \leq$	**4.714**	18.057

CPU time in seconds to find all the solutions

Experiments

- Problems from the MinLPLib benchmark
- Same configuration

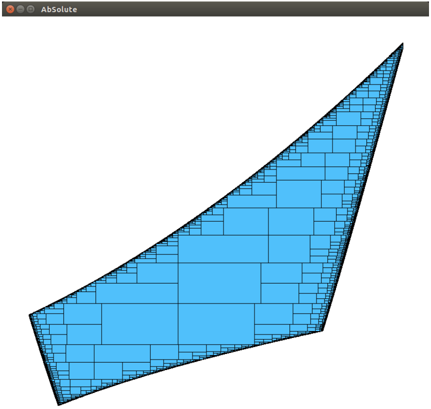
Results

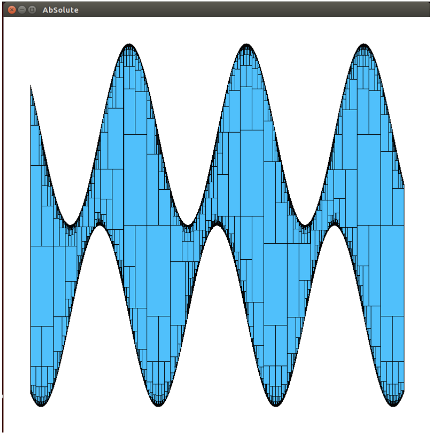
problem	#var	#ctrs	type	boxes	prod
mickey	2	2	$\sim$	**49.015999**	111.618042
eqlin	3	3	/	130.645037	**4.138947**
hs-f1	2	2	/, $\sim$	4.773140	**2.788067**
octo_hole	2	8	/, $\sim$	858.849049	**510.193110**

CPU time in seconds to find all the solutions

Visualization

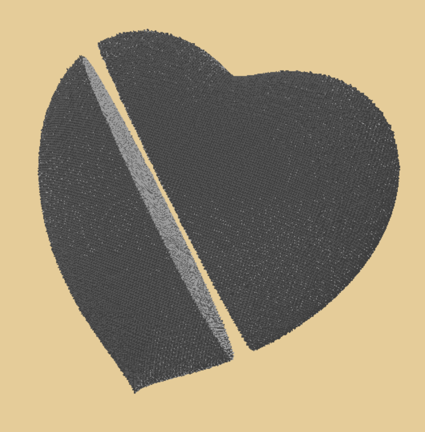
In 2D, problems with only two variables or projection on two variables

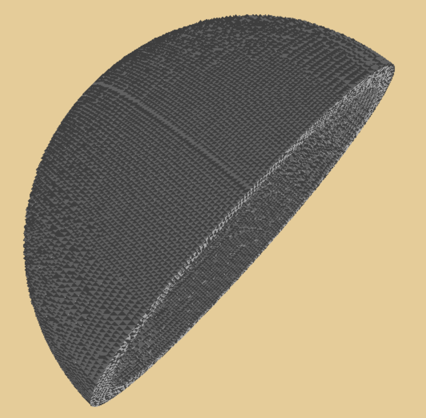




Visualization

In 3D, problems with three variables





Conclusion

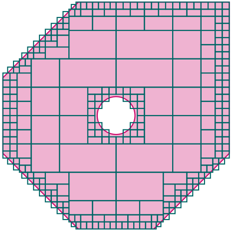
- CP solving method can be defined with AI tools and techniques
- Abstract solving method is modular
- Hybrid CP–AI solver naturally handles mixed constraint problems

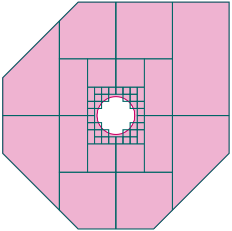
- Need to implement advanced CP heuristics in AbSolute

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?

Chabert, G. and Jaulin, L. (2009).
Contractor programming.
Artificial Intelligence, 173:1079–1100.

Collavizza, H. and Rueher, M. (2007).
Exploring different constraint-based modelings for program verification.
In Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP'07), volume 4741 of Lecture Notes in Computer Science, pages 49–63. Springer.

Cousot, P. and Cousot, R. (1976).
Static determination of dynamic properties of programs.
In Proceedings of the 2nd International Symposium on Programming, pages 106–130.

Cousot, P. and Cousot, R. (1979).
Systematic design of program analysis frameworks.

In Proceedings of the 6th ACM SIGACT-SIGPLAN symposium of Principles of Programming Languages, pages 269–282.

Grimes, D. and Hebrard, E. (2011).
Models and strategies for variants of the job shop scheduling problem.
In Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11), volume 6876 of Lecture Notes in Computer Science, pages 356–372.
Springer-Verlag.

Hervieu, A., Baudry, B., and Gotlieb, A. (2011).
Pacogen: Automatic generation of pairwise test configurations from feature models.
In Proceedings of the 22nd International Symposium on Software Reliability Engineering, pages 120–129.

Jeannet, B. and Miné, A. (2009).
Apron: A library of numerical abstract domains for static analysis.

In Proceedings of the 21th International Conference Computer Aided Verification (CAV 2009).

Lazaar, N., Gotlieb, A., and Lebbah, Y. (2012).
A cp framework for testing cp.
Constraints, 17(2):123–147.

Montanari, U. (1974).
Networks of constraints: Fundamental properties and applications to picture processing.
Information Science, 7(2):95–132.

Pelleau, M., Miné, A., Truchet, C., and Benhamou, F. (2013).
A constraint solver based on abstract domains.
In Proceedings of the 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2013).

Pelleau, M., Truchet, C., and Benhamou, F. (2011).
Octagonal domains for continuous constraints.

In Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11), volume 6876 of Lecture Notes in Computer Science, pages 706–720. Springer-Verlag.

Pelleau, M., Truchet, C., and Benhamou, F. (2014).
The octagon abstract domain for continuous constraints.
Constraints, 19(3):309–337.

Perriquet, O. and Barahona, P. (2009).
Constraint-based strategy for pairwise rna secondary structure prediction.
In Proceedings of the 14th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence (EPIA '09), volume 5816 of Lecture Notes in Computer Science, pages 86–97. Springer-Verlag.

Ponsini, O., Michel, C., and Rueher, M. (2011).

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

Ramamoorthy, V., Silaghi, M. C., Matsui, T., Hirayama, K., and Yokoo, M. (2011).
The design of cryptographic s-boxes using csps.
In Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11), volume 6876 of Lecture Notes in Computer Science, pages 54–68.
Springer-Verlag.

Scott, J. (2016).
Other Things Besides Number: Abstraction, Constraint Propagation, and String Variables.
PhD thesis, University of Uppsala.

Truchet, C. and Assayag, G., editors (2011).

Constraint Programming in Music.
ISTE.