AbSolute, a Constraint Solver based on Abstract Domains

https://github.com/mpelleau/AbSolute

August 23, 2018

Contents

1 Introduction **2**
- 1.1 Authors and Acknowledgements 3
- 1.2 Getting Help 3

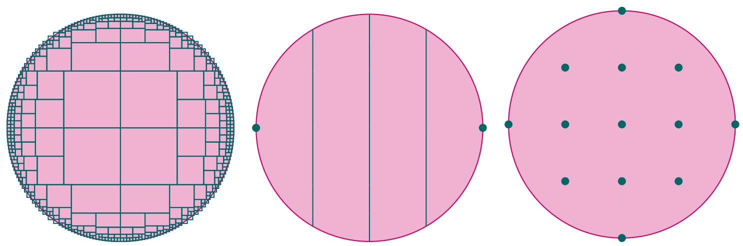
2 Building **3**
- 2.1 From Opam 4
- 2.2 From source 4
- 2.3 Licenses and Copyright 4

3 Getting Started **5**
- 3.1 Solving 5
- 3.2 Syntax 6
  - 3.2.1 Variables 6
  - 3.2.2 Constraints 6
- 3.3 Solving options 7

1 Introduction

AbSolute is a constraint solver based on abstract domains. It implements the solving method presented in [Pelleau et al., 2013] that can be found here `https://hal.archives-ouvertes.fr/hal-00785604/file/Pelleau_Mine_Truchet_Benhamou.pdf`. It relies on Apron [Jeannet and Miné, 2009] an abstract domains library in OCaml.

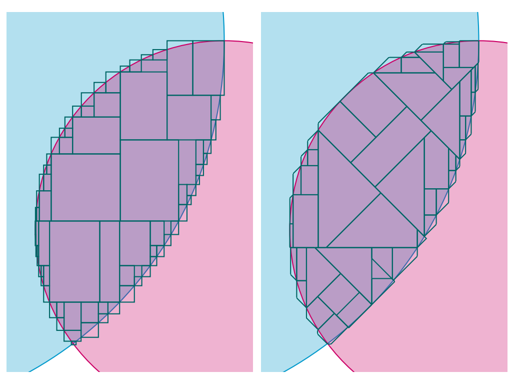
It can be used to solve problems containing real variables, integer variables or both. Figure gives an example of the same constraint and the solutions obtained given the type of variables.



(a) Two reals (b) A real, an integer (c) Two integers

Figure 1: Same constraint on different types of variables

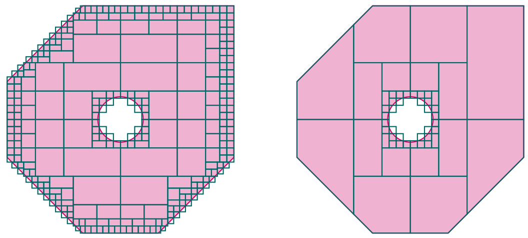
It can also be used to solve problems using non-Cartesian representations. Figure shows the solutions obtained using the boxes or the octagons.



(a) Boxes (b) Octagons

Figure 2: Comparison between boxes solving and octagons solving

Finally, it can also solve using reduced product, like in Figure the problem is solved using the reduced product of boxes and polyhedron.



(a) Boxes (b) Boxes and Polyhedron

Figure 3: Comparison between boxes solving and, boxes and polyhedron product solving

1.1 Authors and Acknowledgements

The bulk of the AbSolute system and its documentation was written by Marie Pelleau. Members of the main team are Ghiles Ziat, who is responsible, for example, for the refactoring of the solver and the visualization tool, Alexandre Marechal, who added the vpl domain and created the opam package. Furthermore, occasional contributions have been made by Antoine Miné and Charlotte Truchet.

The research and the development of the solver has been partly funded by the Coverif ANR project 15-CE25-0002-03.

1.2 Getting Help

When you need help with AbSolute, please do the following:

1. Read the manual, at least the part that has to do with your problem.
2. If that does not solve the problem, try having a look at the GitHub development page (see the title of this document). Perhaps someone has already reported a similar problem and someone has found a solution. Do not hesitate to add an issue on the GitHub repository.
3. As a last resort you can try to email me (Marie Pelleau). I do not mind getting emails, but I cannot guarantee that your emails will be answered timely.

2 Building

Apron[1] is mandatory to build AbSolute. We strongly recommend to install it using the package manager Opam[2].
If you want to use VPL [3] please install it beforehand using the following lines:

[1] `http://apron.cri.ensmp.fr/library/`
[2] `https://opam.ocaml.org`
[3] `https://github.com/VERIMAG-Polyhedra/VPL`

```
opam repo add vpl https://raw.githubusercontent.com/VERIMAG-Polyhedra/opam-vpl/master
opam install vpl-core
```

Here is the list of commands to install AbSolute.

2.1 From Opam

```
opam repo add absolute https://raw.githubusercontent.com/mpelleau/AbSolute/master
opam install absolute
```

Warning For some reason, having both packages libapron and libapron-dev installed (with for instance apt) will make the building of AbSolute fail. Therefore, the easiest way to deal with Apron is to let Opam do the job.

Warning For some reason, on Linux Mint OS, Opam does not seems to install the gmp or mpfr libraries required by Apron. Therefore, the easiest way is to use apt to install gmp and mpfr before and then install Apron using Opam.

```
sudo apt-get install libgmp-dev libmpfr-dev
```

2.2 From source

A simple make in the AbSolute folder will do the job.

2.3 Licenses and Copyright

To reference the AbSolute solver, please cite [Pelleau et al., 2013]

3 Getting Started

3.1 Solving

In Constraint Programming, a problem is formalized under the form of a CSP. In AbSolute, the CSP is described in a text file (see folder problems/ for more examples).

Example: Modelization in AbSolute. Consider the CSP on the integer variable x, and the real variable y with domains $D_x = [\![0, 4]\!]$, $D_y = [0, 4]$, and with the circle constraint $(x-2)^2 + (y-2)^2 <= 4$.

It is translate in AbSolute as:

```
init {
  int x = [0;4];
  real y = [0;4];
}

constraints {
  (x-2)^2 + (y-2)^2 <= 4;
}
```

The `init` part corresponds to the creation of the variables. They are created using their name and domain. Then the constraints are written in the `constraints` part.

AbSolute also handles constants, in the previous example if x is a constant equal to 4, it can be translate in Absolute as:

```
constants {
  x = 4;
}

init {
  real y = [0;4];
}

constraints {
  (x-2)^2 + (y-2)^2 <= 4;
}
```

If you have an optimization problem, the objective function should also be specified in the text file.

Example: Optimisation Problem in AbSolute. Consider the CSP in the previous example with the objective function $x + y$ to minimize.

It is translate in AbSolute as:

```
init {
  int x = [0; 4];
  real y = [0; 4];
}

objective {
  x + y
}

constraints {
  (x-2)^2 + (y-2)^2 <= 4;
}
```

The text file is the same, except for the `objective` part that is added.

Once the problem is described in a text file, the problem can be solved using the command `./absolute problem.abs`. Several examples are given on GitHub, in the problems directory.

3.2 Syntax

We describe here some of the syntax available to describe the problem.

3.2.1 Variables

If a bound of a variable is unknown, then the domain can be described using the infinity symbol `oo`. For example:

```
real x = [-oo; oo];
int y = [-10; oo];
real z = [-oo; 10];
```

3.2.2 Constraints

The usual arithmetic operations are available (`+, -, *, /`). In addition trigonometric functions are available `cos, sin, tan , cot, asin, acos, atan, acot`, and also the following functions `sqrt, ^, exp, ln, log`.

The constraints can be equalties (`=`), disequalities (`!=`), inequalities (`<, >, <=, >=`).

```
y <= cos(x);
z = sqrt(x + y);
t > ln(z);
w != x^2 + y^2;
```

By default the constraints considered in the `constraints` part formed a conjunction, if you want to specify a disjunction, you can do so using the symbol `||`, and if needed the conjonction symbol is `&&`.

```
constraints {
  y<x || (y>-x && x >= z);
}
```

3.3 Solving options

Several options exist, there are listed in the following table.

Search parameters	
`-minimize` or `-m`	Specify that the problem is a minimization problem
`-precision` or `-p` *value*	Changes the precision for *value*, default 1e-3
`-max_sol` *value*	Changes the maximum number of solutions, default 1e6
`-max_iter` *value*	Changes the maximum number of iterations, default 1e7
`-sure` or `-s`	Keeps only the sure solutions
`-iter` or `-i`	Enables the loop for the propagation
`-no-rewrite`	Disables the constraint rewriting, enabled by default
`-pruning`	Enables the "pruning" during the solving process
`-pruning_iter` or `-pi` *value*	Changes the number of times the pruning process is applied
`-split` or `-sp` *split*	Changes the splitting strategy used for the solving Possible values are default, maxSmear, smear.

Domains parameters	
`-domain` or `-d` *dom*	Changes the domain used for the solving, default box The possible values for *dom* are: box: box with floating-point bounds abstract domain boxQ: box with rational bounds abstract domain boxS: box with floating-point bounds and strict inequalities abstract domain boxQS: box with rational bounds and strict inequalities abstract domain oct: octagon abstract domain poly: polyhedron abstract domain vpl: convex polyhedron abstract domain

`-lin`	Sets the linearization algorithm of the VPL
`-vpl_split`	Sets the split strategy of the VPL

Visualization parameters	
`-tex`	Prints the solutions in latex format on standard output
`-trace` or `-t`	Prints the solutions on standard output
`-visualization` or `-v`	Enables visualization mode
`-obj`	Generates an .obj file (for 3D visualization)
`-sbs`	Enabling step by step visualization

Miscelleanous	
`-debug`	Prints the execution for debug purpose
`-debug_lv`	Set the debug level. The higher, most print you get
`-help` or `--help`	Display this list of options

Examples of usage

```
./absolute problems/poly_hole.abs -v
./absolute problems/poly_hole.abs -v -pruning
./absolute problems/poly_hole.abs -d poly -v
```

References

[Jeannet and Miné, 2009] 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)*.

[Pelleau et al., 2013] 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)*.