AbSolute, a Constraint Solver based on Abstract Domains

`https://github.com/mpelleau/AbSolute`

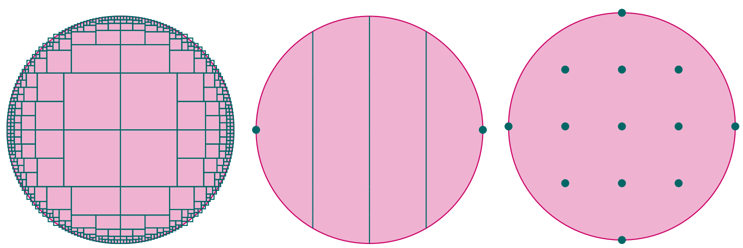
December 15, 2017

Contents

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.

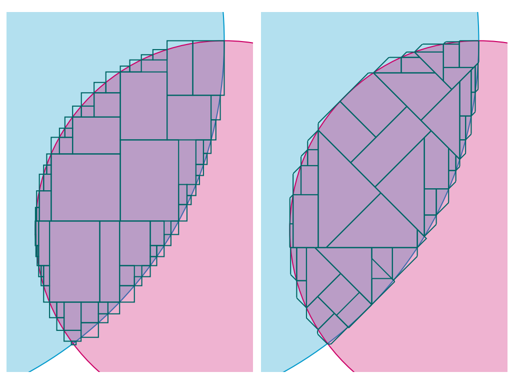
It can be used to solve problems containing real variables, integer variables or both. Figure gives an example of the same constraint and the solutions obtained given the type of variables.



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

Figure 1: Same constraint on different types of variables

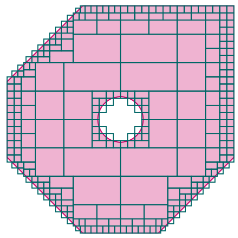
It can also be used to solve problems using non-Cartesian representations. Figure shows the solutions obtained using the boxes or the octagons.



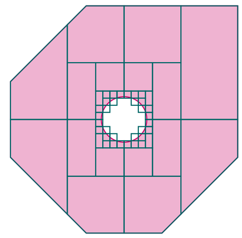
(a) Boxes (b) Octagons

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. A further member of the main team is Ghiles Ziat, who is responsible, for example, for the refactoring of the solver and the visualization tool. Furthermore, occasional contributions have been made by Antoine Miné.

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.
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] and Zarith[2] are mandatory to build AbSolute. We strongly recommend to install it using the package manager Opam[3]. Then a simple `make` will do the job. Here is the list of commands to install AbSolute using Opam.

```
opam switch 4.05.0
eval `opam config env`
```

[1] http://apron.cri.ensmp.fr/library/
[2] https://github.com/ocaml/Zarith
[3] https://opam.ocaml.org

```
opam install apron
opam install zarith
```

Move to the AbSolute folder

```
make
```

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 install it with and only with Opam.

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
```

Then follow the instructions in the Building section

2.1 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.

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 `constraint` 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 Example 3.1 with the objective function $x + y$ to minimize

It is translate in AbSolute as:

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 `./solver.opt problem.abs`. Several examples are given on GitHub, in the problem directory.

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

objective {
  x + y
}

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

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);
```

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 $||$.

```
constraints {
  y<x || y>-x;
}
```

3.3 Solving options

Several options exist, there are listed in this section.

-visualization or -v	Enables visualization mode
-minimize or -m	Specify that the problem is a minimization problem
-precision or -p *value*	Changes the precision for *value*, default 1e-3
-domain or -d *dom*	Changes the domain used for the solving, default box
	The possible values for *dom* are:
	box: box abstract domain
	oct: octagon abstract domain
	poly: polyhedron abstract domain
	boxNoct: reduced product of boxes and octagons
	boxNpoly: reduced product of boxes and polyhedra
	octNpoly: reduced product of octagons and polyhedra
-trace or -t	Prints the solutions on standard output
-sure or -s	Keeps only the sure solutions
-pruning	Enables the "pruning" during the solving process
-max_sol *value*	Changes the maximum number of solutions, default 1e6
-max_iter *value*	Changes the maximum number of iterations, default 1e7
-tex	Prints the solutions in latex format in a file
-obj	Generates an .obj file (for 3D visualization)
-help or –help	Display this list of options

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)*.