ConArg: A Constraint-programming Solver for Abstract Argumentation Problems

Stefano Bistarelli, Fabio Rossi, Francesco Santini, Carlo Taticchi
Department of Mathematics and Computer Science
University of Perugia
firstname.lastname@unipg.it

Abstract

ConArg is a Constraint Programming (CP) solver dedicated to the solution of problems related to extension-based semantics in Abstract Argumentation. It exploits Gecode, an efficient C++ toolkit for developing constraint-based systems and applications. The properties required by semantics are encoded as constraints, and arguments are assigned to *true* if belonging to a valid extension for that semantics. The search for solutions (as enumerating extensions or checking argument-acceptance) takes advantage of well-known techniques in CP, as local consistency, different variable and value heuristics, and a complete exploration of the solution space with branch-and-bound pruning.

Description

ConArg (*Arg*umentation with *Con*straints) is a Constraint-programming tool oriented to the solution of problems related to extension-based semantics in Abstract Argumentation [9]. Since the first version of the tool [2, 6], it has been updated with the purpose *i)* to solve further problems linked to weighted problems [4] and coalitions of arguments [7], and *ii)* to improve its performance over classical semantics, by using a benchmark assembled with random graph-models [3]. The main design principles consists in ensuring correctness of solutions and solving weighted extensions of Abstract Argumentation.

The first version of ConArg [6] was based on the *Java Constraint Programming* solver[1] (*JaCoP*), a Java library that provides a *Finite Domain Constraint Programming* paradigm [11]. The current version of ConArg exploits *Gecode 6.2.0*[2], an efficient C++ toolkit for developing constraint-based systems and applications. ConArg is now implemented also as a C++ software library [5], which can be used in programs to compute extensions and use them in decision-making applications, for example. ConArg and ConArgLib are among the official projects supported by Gecode.[3]

In [8] the authors classify the ConArg approach among "reduction-based implementations": these methods first reduce the problem to the target formalism (in this case, constraints), then run the solver of the target formalism, and finally interpret the output as the solutions of the original problem; other similar approaches use Answer Set Programming or SAT solvers.

In ConArg, the search procedure takes advantage of classical techniques, such as local consistency, different heuristics for trying to assign values to variables, and complete search-tree with branch-and-bound. Models in Gecode are implemented using *spaces*. A space is home to *variables*, *propagators* (implementations of constraints), and *branchers* (implementations of branching, describing shape of the search tree).

An array of Boolean variables *args[ ]* (i.e., instances of the class *BoolVar*) represents the whole set of arguments $\mathcal{A}_{rgs}$; Boolean variables can only take 0 or 1 values. An array of Boolean variables can be created with *BoolVarArray args[ ](space,* $|\mathcal{A}_{rgs}|$*, 0, 1)*, where *space* is the associated search-space. For each modelled constraint there is *post* function (*rel* in the following examples) that creates propagators implementing the constraint in the home space, passed

[1] http://www.jacop.eu.
[2] http://www.gecode.org.
[3] Gecode projects: https://www.gecode.org/projects.html.

as argument. As an example, constraints for modelling conflict-free-sets are based on the first order logic formula $\forall a, b \in \mathcal{A}_{arg}$ s.t. $R(a, b)$, then $a \Rightarrow \neg b$ ($>>$ is the implication operator in Gecode): $rel(space, args[i] >> !args[j])$. For a more detailed description of constraint encoding, we point the interested reader to [5].

ConArg was submitted to the first *International Competition on Computational Models of Argumentation* (ICCMA 2015) [12] and ICCMA 2017 [10]. It was the reference solver in ICCMA 2019, used to check the correctness of solutions provided by participants [1].

The version of ConArg we submitted to ICCMA 2021 participate in the following *classical* tracks:

- **CE**: counting the number of extensions of one complete, preferred, stable, semi-stable and stage semantics.
- **SE**: returning one extension given one complete, preferred, stable, semi-stable, ideal and stage extensions semantics;
- **DC**: checking the credulous acceptance for the complete, preferred, stable, semi-stable and stage semantics;
- **DS**: checking the sceptical acceptance for the complete, preferred, stable, semi-stable, ideal and stage extensions semantics.

From the home-page of ConArg[4], it is possible to download the executable of the solver, compiled for Linux i386 and x64 machines. Moreover, still at the same Website, we offer a visual interface where to interactively draw abstract frameworks (arguments and attacks as directed edges), and use ConArg as the underlying solver for the requested problem. Finally, the version used as the reference solver in ICCMA 2019 can be pulled as a Docker image.[5]

References

[1] S. Bistarelli, L. Kotthoff, F. Santini, and C. Taticchi. A first overview of iccma'19. In *Proceedings of the Workshop on Advances In Argumentation In Artificial Intelligence 2020 co-located with the 19th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2020)*, volume 2777 of *CEUR Workshop Proceedings*, pages 90–102. CEUR-WS.org, 2020.

[2] S. Bistarelli, D. Pirolandi, and F. Santini. Solving weighted argumentation frameworks with soft constraints. In *ERCIM International Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)*, volume 6384 of *LNCS*, pages 1–17, 2009.

[3] S. Bistarelli, F. Rossi, and F. Santini. Not only size, but also shape counts: abstract argumentation solvers are benchmark-sensitive. *J. Log. Comput.*, 28(1):85–117, 2018.

[4] S. Bistarelli, F. Rossi, and F. Santini. A novel weighted defence and its relaxation in abstract argumentation. *Int. J. Approx. Reason.*, 92:66–86, 2018.

[5] S. Bistarelli, F. Rossi, and F. Santini. Conarglib: an argumentation library with support to search strategies and parallel search. *Journal of Experimental & Theoretical Artificial Intelligence*, 0(0):1–28, 2020.

[6] S. Bistarelli and F. Santini. Conarg: A constraint-based computational framework for argumentation systems. In *23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 605–612. IEEE Computer Society, 2011.

[7] S. Bistarelli and F. Santini. Coalitions of arguments: An approach with constraint programming. *Fundam. Inform.*, 124(4):383–401, 2013.

[8] F. Cerutti, S. A. Gaggl, M. Thimm, and J. P. Wallner. Foundations of implementations for formal argumentation. In P. B. D. G. M. G. L. van der Torre, editor, *Handbook on Formal Argumentation*, pages 688–767. College Publications, February 2018.

[9] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–357, 1995.

[10] S. A. Gaggl, T. Linsbichler, M. Maratea, and S. Woltran. Design and results of the second international competition on computational models of argumentation. *Artif. Intell.*, 279, 2020.

[11] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[12] M. Thimm, S. Villata, F. Cerutti, N. Oren, H. Strass, and M. Vallati. Summary report of the first international competition on computational models of argumentation. *AI Magazine*, 37(1):102, 2016.

[4] `http://www.dmi.unipg.it/conarg/`.

[5] A Docker image of ConArg used in ICCMA'19`https://hub.docker.com/r/iccma19/conarg`.