ConArg2: A Constraint-based Tool for Abstract Argumentation

Stefano Bistarelli[1], Fabio Rossi[1], Francesco Santini[2]

[1] Dipartimento di Matematica e Informatica, Università di Perugia
[bista,rossi]@dmi.unipg.it
[2] Istituto di Informatica e Telematica, CNR, Pisa
francesco.santini@iit.cnr.it

Abstract. ConArg2 is a Constraint-programming tool oriented 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 into constraints, and arguments are assigned to 1 (i.e., *true*) if belonging to a valid extension for that semantics. Searching for solutions of problems (as enumerating extensions or checking argument-acceptance) takes advantage of well-known techniques as local consistency, different heuristics for trying to assign values to variables, and complete search-tree with branch-and-bound.

Description

ConArg (Argumentation with Constraints) is a Constraint-programming tool oriented to the solution of problems related to extension-based semantics in Abstract Argumentation [10]. Since the first versions of the tool [1,7], we have updated it with the purpose *i)* to solve further problems linked to weighted problems [6] and coalitions of arguments [9], and *ii)* to improve its performance over classical semantics, by using a benchmark assembled with random graph models [2,3,4,5]. The first version of ConArg [7,8] is based on the *Java Constraint Programming* solver[3] (*JaCoP*), a Java library that provides a *Finite Domain Constraint Programming* paradigm [12]. The tool comes with a graphical interface, which allows the user to browse all the obtained extensions.

For the sake of performance, we have developed a second version of the tool, i.e., *ConArg2*, which has been submitted to the *International Competition on Computational Models of Argumentation* (ICCMA 2015)[4]. ConArg2 exploits *Gecode 4.4.0*[5], an efficient C++ toolkit for developing constraint-based systems and applications. The properties of semantics are encoded into constraints, and arguments are assigned to 1 (*true*) if belonging to a valid extension for that semantics (0 otherwise). Searching for solutions takes advantage of classical

[3] http://www.jacop.eu
[4] http://argumentationcompetition.org/index.html
[5] http://www.gecode.org

techniques, such as local consistency (through constraint propagation), different heuristics for trying to assign values to variables, and complete search-tree with branch-and-bound. We have also dropped the graphical interface of the first Java system, having a textual output only.

ConArg2 can be currently used to

- enumerate all conflict-free, admissible, complete, stable, grounded, preferred, semi-stable, ideal, and stage extensions;
- return one extension given one of the semantics above;
- check the credulous and sceptical acceptance for the conflict-free, admissible, complete, and stable semantics;
- find the α-semantics described in [6].

From the home-page of ConArg[6], it is possible to download both ConArg2 and ConArg (Java version). Moreover, we offer a visual Web-interface where to draw abstract frameworks (arguments and attacks as directed edges), and then solve some of the problems above. From the home-page it is possible to download ConArg2 compiled for Linux i386 and x64 machines.

The basic command-line usage is described in Fig. 1. Some practical examples are: to enumerate all admissible extensions: *"conarg_gecode -e admissible file.dl"*, to check the sceptical acceptance of argument "a" with the stable semantics *"conarg_gecode -e stable -s a file.dl"*, to compute all α-complete extensions [6] with $\alpha = 3$ *"conarg_gecode -e a-complete -a 3 file.dl"*. An input file *file.dl* follows the ASPARTIX format [11]: e.g., *arg(a)* for defining argument a, and $att(a, b)$ for declaring an attack from a to b.

We briefly show how we map AAFs to *Constraint Satisfaction Problems* (CSPs) [12] in ConArg2. A CSP can be defined as a triple $P = \langle V, D, C\rangle$, where C is a set of constraints defined over the variables in V, each with domain D. Given a framework $\langle A, R\rangle$, we define a variable for each argument $a_i \in A$ ($V = \{a_1, a_2, \ldots, a_n\}$) and each of these arguments can be taken or not in an extension, i.e., the domain of each variable is $D = \{1, 0\}$. As an example we report conflict-free and stable constraints, which can be respectively used to model the conflict-free and (in combination) stable semantics.

- Conflict-free constraints. If $R(a_i, a_j)$ is in the framework we need to prevent a solution to include both a_i and a_j: $\neg(a_i = 1 \wedge a_j = 1)$. All other possible variable assignments $(a = 0 \wedge b = 1)$, $(a = 1 \wedge b = 0)$ and $(a = 0 \wedge b = 0)$ are permitted.
- Stable constraints. If we have a node a_i with multiple parents (in the Argumentation graph) $a_{f1}, a_{f2}, \ldots, a_{fk}$, we need to add a constraint $\neg(a_i = 0 \wedge a_{f1} = 0 \wedge \cdots \wedge a_{fk} = 0)$. In words, if a node is not taken in an extension (i.e. $a_i = 0$), then it must be attacked by at least one of the taken nodes, that is at least a parent of a_i needs to be taken in a solution (that is, $a_{fj} = 1$). Moreover, if a node a_i has no parent in the graph, it has to be included in every extension, i.e., $\neg(a_i = 0)$.

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

```
USAGE:
conarg_gecode  [-s <string> ] [-c <string> ] [-a <double> ] -e <string>
[--] [--version] [-h] <string>

Where:
-s <string> ,  --skeptical <string>
Test an argument for sceptical acceptance (conflict-free, admissible,
     complete and stable semantics only).
-c <string> ,  --credulous <string>
Test an argument for credulous acceptance (conflict-free, admissible,
     complete and stable semantics only).
-a <double> ,  --alpha <double>
Alpha consistency budget (for alpha extensions only).
-e <string> ,  --extension <string>
(required)  Extensions to be enumerated (conflict-free,  admissible,
     complete, stable, preferred, grounded, semi-stable, ideal, stage, a-
     conflict-free,  a-admissible, a-complete, a-stable, a-preferred, a-
     grounded semantics).
--,  --ignore_rest
Ignores the rest of the labeled arguments following this flag.
--version
Displays version information and exits.
-h,  --help
Displays usage information and exits.
<string>
(required)  Input File  in Aspartix format.
```

Fig. 1. How to call ConArg2 from command-line.

Preferred extensions are found by assigning as more arguments as possible to 1 while searching for complete extensions. For this we use the Gecode heuristics *INT_VAL_MAX* (such value is always 1 in our model).

Given a semantics, the credulous acceptance for an argument a is checked by setting that argument to 1 and then halting as soon as an extension containing a is found (i.e., a is credulously accepted). In the worst case, all the search tree is explored without any result, i.e., a is *not* credulously accepted. Checking the sceptical acceptance is a dual problem: given a semantics, we set a to 0 and then we stop as soon as an extension containing a is found (i.e., a is *not* credulously accepted). In the worst case, all the search tree is explored without any result, i.e., a is sceptically accepted.

From the tests and comparisons we perform in [2,3,4,5], we obtain that ConArg behaves fast on lower-order semantics (admissible, complete, and stable ones). Moreover, we notice that our approach proves to be more efficient on some graph topologies than others. For instance, we deal with Barabasi-Albert random graph-models (and trees) better than Kleinberg or Erdős-Rényi models, considering the same *nodes*/*edges* ratio.

In the future we would like to extend ConArg2 to solve coalition-based problems [9], and labelling-based extensions, where having an assignment domain wider than just $\{true, false\}$ suggests the use of a constraint-based solver. Further possible extensions concern Bipolar Argumentation Frameworks, or Constrained-Argumentation Frameworks, where additional used-defined constraints can be adopted to select only some extensions of a given semantics (e.g., "when a is in, then also b must be in"). In addition, we are currently exploring applications of our tool as a reasoning engine for Cybersecurity problems and Decision-making.

Acknowledgement. We thank the following people who provided help during the development of the ConArg project: Carlo Taticchi, Luca Tranfaglia, Paola Campli, and Daniele Pirolandi.

References

1. 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.
2. S. Bistarelli, F. Rossi, and F. Santini. Benchmarking hard problems in random abstract AFs: The stable semantics. In *Computational Models of Argument - Proceedings of COMMA*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 153–160. IOS Press, 2014.
3. S. Bistarelli, F. Rossi, and F. Santini. Efficient solution for credulous/sceptical acceptance in lower-order dung's semantics. In *26th IEEE International Conference on Tools with Artificial Intelligence, (ICTAI)*, pages 800–804. IEEE Computer Society, 2014.
4. S. Bistarelli, F. Rossi, and F. Santini. Enumerating extensions on random abstract-AFs with ArgTools, Aspartix, ConArg2, and Dung-O-Matic. In *Computational Logic in Multi-Agent Systems - 15th International Workshop, CLIMA XV*, volume 8624 of *Lecture Notes in Computer Science*, pages 70–86. Springer, 2014.
5. S. Bistarelli, F. Rossi, and F. Santini. A first comparison of abstract argumentation reasoning-tools. In *ECAI 2014 - 21st European Conference on Artificial Intelligence*, volume 263 of *FAIA*, pages 969–970. IOS Press, 2014.
6. S. Bistarelli and F. Santini. A common computational framework for semiring-based argumentation systems. In *ECAI 2010 - 19th European Conference on Artificial Intelligence*, volume 215 of *FAIA*, pages 131–136. IOS Press, 2010.
7. 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.
8. S. Bistarelli and F. Santini. Modeling and solving AFs with a constraint-based tool: Conarg. In *Theory and Applications of Formal Argumentation (TAFA)*, volume 7132 of *LNCS*, pages 99–116. Springer, 2012.
9. S. Bistarelli and F. Santini. Coalitions of arguments: An approach with constraint programming. *Fundam. Inform.*, 124(4):383–401, 2013.
10. 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.
11. U. Egly, S. A. Gaggl, and S. Woltran. Answer-set programming encodings for argumentation frameworks. *Argument & Computation*, 1(2):147–177, 2010.
12. F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence).* Elsevier Science Inc., New York, NY, USA, 2006.