Computational Models of Argument
B. Verheij et al. (Eds.)
IOS Press, 2012

doi:10.3233/978-1-61499-111-3-505

Evaluating Abstract Dialectical Frameworks with ASP [1]

Stefan Ellmauthaler and Johannes Peter Wallner
Institute of Information Systems, Vienna University of Technology, Austria

A widespread modeling language for abstract argumentation is the argumentation framework due to Dung [1]. However relating arguments in this framework is solely based on the notion of direct binary attacks. More sophisticated relations require auxiliary structures tailored to the chosen semantics. We present a system called ADFsys based on the more general *abstract dialectical frameworks* (ADFs) [2] to overcome this technical necessity.

The idea of ADFs is to apply the expressiveness of propositional logic to represent relations between arguments and hence generalize argumentation frameworks. Each argument comes with an acceptance condition, which states when it can be accepted. This condition refers to the status of the other arguments. Figure 1 shows an example ADF, where the argument a can only be accepted if not both b and d are accepted. The arrows represent dependencies between the arguments.

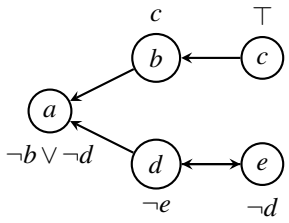

Figure 1. Abstract dialectical framework

To complement theoretical results, practical implementations are required for experimentation and to deepen the understanding of the different semantics and their applicability. We implemented the ADFsys using the answer-set-programming (ASP) paradigm with the solver *clasp* [3] to achieve a platform independent solution. This reduction to ASP has the benefit of delegating the burden of computation to highly sophisticated and well developed systems. This approach is in line of the successful existing approach of ASPARTIX [4], which utilizes ASP for argumentation framework problems. ADFsys takes an ADF as the input and outputs the sets of accepted arguments according to the chosen semantics. Readily available in the system are the implementations of the conflict-free and admissible concepts, as well as the model, stable, preferred and well-founded semantics.

[1]This work has been funded by Vienna Science and Technology Fund (WWTF) through project ICT08-028

The input language is in the style of answer-set-programming. The example from Figure 1 can be encoded as follows, where statement(x) specifies that x is an argument and ac(x, f) denotes the acceptance condition f of x. The condition f now refers to statements as atoms and may consist of the usual boolean operators, arbitrarily nested: and, or and neg. The truth constants are represented by c(v) and c(f) for true and false respectively. To reduce the complexity and therefore optimize the performance of certain computations (i.e. the admissible concept, as well as the stable and preferred semantics), it is possible to give additional information on the relation between the statements.

$$\begin{array}{ll} \text{statement}(a). & \text{ac}(a, \text{or}(\text{neg}(b), \text{neg}(c))). \\ \text{statement}(b). & \text{ac}(b, c). \\ \text{statement}(c). & \text{ac}(c, \text{c}(\text{v})). \\ \text{statement}(d). & \text{ac}(d, \text{neg}(e)). \\ \text{statement}(e). & \text{ac}(e, \text{neg}(d)). \end{array}$$

In the following we show the partial ASP module π_{mod} to compute ADF models. The first two lines make a guess for each argument if it is in or out of the model. The acceptance conditions are calculated via a bottom-up evaluation under the current guess. The predicates ismodel and nomodel represent the result, i.e. a satisfied or an unsatisfied condition respectively. This is handled by another ASP module, which is omitted here due to lack of space. The two constraints in the last two lines now make sure that the condition of every selected argument evaluates to true and for all other arguments to false.

$$\begin{aligned} \pi_{mod} = \{ \quad & \text{in}(X) \leftarrow \mathit{not}\, \text{out}(X), \text{statement}(X). \\ & \text{out}(X) \leftarrow \mathit{not}\, \text{in}(X), \text{statement}(X). \\ & \leftarrow \text{in}(X), \text{ac}(X, F), \text{nomodel}(F). \\ & \leftarrow \text{out}(X), \text{ac}(X, F), \text{ismodel}(F). \ \} \end{aligned}$$

The system is available at `www.dbai.tuwien.ac.at/research/project/argumentation/adfsys`

References

[1] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[2] Gerhard Brewka and Stefan Woltran. Abstract Dialectical Frameworks. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*, pages 102–111. AAAI Press, 2010.

[3] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24(2):105–124, 2011.

[4] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument and Computation*, 1(2):147–177, 2010.