Applications of ASP in Formal Argumentation

Martin Diller, Wolfgang Dvořák, Jörg Pührer,
Johannes P. Wallner, and Stefan Woltran

Institute of Logic and Computation, TU Wien, Austria

Abstract. Argumentation is nowadays one of the major fields within Artificial Intelligence. On one hand it provides genuine methods to model discourses or legal cases; on the other hand, it is closely related to – and gives an orthogonal view on – several formalisms from the AI domain. The aim of this paper is to survey argumentation systems that use ASP technology inside.

Keywords: Argumentation, Answer-Set Programming

1 Introduction

The area of formal argumentation has seen a steady rise of interest in the last two decades in the field of Artificial Intelligence [5]. It studies how to model arguments and their relationships, as well as the necessary conflict resolution in the presence of diverging opinions, thus providing a general and formal treatment of several questions arising in various applications, for instance, in legal reasoning. In its most prominent manifestation, Abstract Argumentation Frameworks (AFs) by Dung [23], argumentation scenarios are modeled as simple directed graphs, where the vertices represent arguments and each edge corresponds to an attack between two arguments. AFs are resolved using appropriate semantics, which select acceptable subsets of the arguments. For example, consider the AF $F = \langle \{a, b, c, d\}, \{\langle a, b\rangle, \langle b, c\rangle, \langle b, d\rangle\}\rangle$ with arguments a, b, c, and d, where a attacks b and b attacks c and d. F has five extensions $\emptyset, \{a\}, \{a, c\}, \{a, d\}, \{a, c, d\}$ under admissible semantics and one extension $\{a, c, d\}$ under complete semantics. Many more semantics have been studied and formal argumentation offers a wide range of different formalisms, ranging from AFs to structured argumentation, where arguments are not treated as atomic entities [8].

Logic Programming (LP) and Argumentation are closely related fields. They have common roots in non-monotonic reasoning and, like in the answer-set programming (ASP) paradigm, many argumentation semantics allow for multiple intended models. There is a research stream that instantiates logic programs as argumentation frameworks with the goal to find semantics for the latter in order to capture the LP semantics at hand. This has been done for stable semantics already in the seminal paper by Dung [23] and has been subsequently applied to other semantics, see e.g. [16]. Another research stream takes the other direction and investigates methods for implementing argumentation by means of logic programming, in particular, by making use of ASP solvers [38,39,40]. The

aim of this paper is to survey these ASP-based approaches. For more general overviews on argumentation systems we refer to [18,19].

2 ASP-Based Argumentation Systems

Dung's Abstract Argumentation Frameworks. AFs are a core formalism in formal argumentation and have been extensively studied in the literature. In a nutshell, AFs abstract away from the contents of single arguments, focusing on the relationship between arguments that is analyzed. The relationship that is expressed by an AF is the so-called attack relation, where an attack from an argument a to argument b indicates that the conclusion of a contradicts (parts of) the premises of an argument b. AFs are thus just directed graphs, where arguments constitute the vertices and edges represent attacks. Different *semantics* are defined to obtain jointly acceptable sets of arguments (the so-called extensions of the AF at hand). Nowadays, a large number of such semantics is available and these semantics were extensively studied, see e.g. [7]. The complexity of reasoning tasks for AFs span different classes on the first two levels of the polynomial hierarchy [24] depending on the actual semantics.

Early approaches to encode AF evaluation via ASP are [48,59,60], see also the survey by Toni and Sergot [58]. The reference ASP system for AFs, however, is nowadays ASPARTIX[1], which provides a collection of ASP encodings for numerous AF semantics [28]. ASPARTIX strictly separates the input AF from the evaluation. In other words, for each semantics, a fixed encoding is provided which, when combined with an AF as input, returns the corresponding extensions. In particular, for semantics which are located on the second level of the polynomial hierarchy, different approaches have been presented which rely on the standard saturation technique in ASP [28,36] or make use of particular features of ASP solvers [25,34]. A different approach to capture the different semantics for AFs is via labellings; ASP encodings based on labellings have been presented in [52].

Formalisms Extending AFs. The quite restricted structure of Dung AFs lead to proposals of several extensions of this formalism. Most prominent among these extensions are collective attacks [47], support relations [17] and attacks on attacks [45] yielding formalisms known as SETAFs, bipolar AFs, and EAFs, respectively. While support relations have already been considered in the ASPARTIX system from the very beginning, there are dedicated ASP encodings available for collective attacks [27] and attacks on attacks[2] [26]. In recent work [42], ASPARTIX was extended to include Metalevel AFs that simultaneously allow for values, preferences and attacks on attacks.

Abstract Dialectical Frameworks. Among the most comprehensive of the generalizations of Dung AFs are abstract dialectical frameworks or ADFs for short

[1] `http://www.dbai.tuwien.ac.at/research/argumentation/aspartix/`
[2] `http://gerd.dbai.tuwien.ac.at/`

[12,13]. In ADFs, each argument comes with an associated acceptance condition in the form of a propositional formula that allows to express how the acceptance status of an argument depends on the acceptance status of the parents in the graph. The main semantics for AFs have been generalized to ADFs, with the higher expressivity of ADFs coming at the price of higher complexity: the complexity of most reasoning tasks for ADFs are one level higher in the polynomial hierarchy with respect to the complexity of the same tasks for AFs [55]. For instance, checking credulous acceptance under admissible semantics is NP-complete for AFs and Σ_2^P-complete for ADFs. Bipolar ADFs on the other hand, which allow only attacking and supporting links between statements, while still being more expressive than AFs, have the same complexity.

The first ASP-based system for ADFs was ADFSys [33] which lead to the DIAMOND (DIAlectical FraMewOrks eNcoDings)[3] family of systems [30,31]. All of these systems have in common that they basically consist of several ASP encodings and a script with information on what encodings to combine to obtain a desired reasoning behavior. From the start, the systems in the DIAMOND family support reasoning with subclasses of ADFs with the last version of DIAMOND, goDIAMOND [32], allowing ADFs with acceptance conditions expressed as propositional formulas and as Boolean functions, as well as bipolar ADFs and AFs (previous versions also included prioritized ADFs). The encoding for the different semantics usually stays the same; what changes for the different subclasses are the encodings for the computation of the consequence operators associated to the different formalisms to obtain the desired semantics.

The different versions of DIAMOND typically rely on static encodings, i.e. the encodings do not change for different ADF instances. This approach is hence limited by the data complexity of ASP. To handle reasoning problems going beyond the second level of the polynomial hierarchy, DIAMOND systems transform ADFs with acceptance conditions expressed as propositional formulas to ADFs with Boolean functions; an operation which may involve an exponential blowup. YADF[4] [11] is a more recent system demonstrating the use of the fact that the combined complexity of ASP for programs with predicates of bounded arity [29], just as the complexity of reasoning of ADFs, reaches the third level of the polynomial hierarchy. This allows for dynamic (hence the "Y" in YADF) yet single shot encodings to fragments of ASP with matching complexity that make use of the representation of the acceptance conditions of the input ADFs as propositional formulas.

The GRAPPA approach. GRAPPA (GRaph-based Argument Processing with Patterns of Acceptance) [14] is a further generalization of ADFs, providing means of semantically evaluating arbitrary labelled graphs with flexible user-defined acceptance conditions. Reasoning for GRAPPA nevertheless is exactly as hard as reasoning for ADFs. GrappaVis[5] [41] is a graphical JAVA-based tool allowing to

[3] `http://diamond-adf.sourceforge.net/`
[4] `https://www.dbai.tuwien.ac.at/proj/adf/yadf/`
[5] `https://www.dbai.tuwien.ac.at/proj/adf/grappavis/`

specify GRAPPA (and ADF) instances, evaluate them and visualize the results. It is based on static as well as dynamic encodings to ASP.

ASP is also used as the back-end of ArgueApply[6] [50,51], a mobile app for online debates based on GRAPPA. ArgueApply is a native Android application where users can participate in discussions and collectively relate statements of the debates with each other. Based on these relationships, a debate can be semantically evaluated on the Android device exploiting translations from GRAPPA to ASP [11] developed for YADF. Reasoning is done by an ASP service that may run independently of the Android application and encapsulates a JavaScript version of Clingo 5.2.

Synthesis and Enforcement Problems. UNREAL[7] [44] is a system that deals with *realizability* of abstract argumentation: the problem of whether for a given set V of interpretations, there is an instance of the argumentation formalisms that has V as its semantics. This task is also known as synthesis problem. UNREAL can decide realizability for multiple formalisms (AFs, ADFs, BADFs, and SETAFs) and semantics. Moreover, if a set is realizable, the tool can compute frameworks that evaluate to V. The tool uses modular ASP encodings that guess semantical objects, called *realizability characterizations*, that determine the realizing knowledge base. Depending on the chosen formalism and semantics, different propagation modules efficiently eliminate unwanted solution candidates.

Enforcement is a dynamic operation on formal models in argumentation. Operators within this family transform a given formal model (e.g. expanding arguments and attacks) such that the modified model satisfies certain goals (for instance, having a specified argument accepted). Encoding such operations in ASP was studied for AFs [49,52] and ADFs [61].

Structured argumentation In formal argumentation, one distinguishes between structured argumentation, where arguments have explicit internal structure, and abstract argumentation, where arguments' internal contents are abstracted away to focus on their acceptability solely relying on relationships between arguments. Usually, structured arguments are constructed from a possibly inconsistent and/or incomplete (formal) knowledge base (e.g. a logical or rule base). Some of the most established frameworks for structured argumentation [9] include deductive argumentation [10], ASPIC+ [46], defeasible logic programming [37], and assumption-based argumentation [21].

In deductive argumentation [10] arguments are inferences based on a monotonic logic, notably classical logic. Different notions of conflict among arguments can been defined, allowing the construction of AFs (the attacks standing for the conflicts) by means of which inconsistencies in the knowledge bases which start off the deductive argumentation process can be resolved. ASP has been utilized to implement argument construction (as well as their relationships) for deductive argumentation in the work of [20] resulting in the tool vispartix[8]. More

[6] `https://www.informatik.uni-leipzig.de/~puehrer/ArgueApply/`
[7] `http://www.dbai.tuwien.ac.at/proj/adf/unreal/`
[8] `http://www.dbai.tuwien.ac.at/research/project/argumentation/vispartix/`

concretely, the approach relies on a two-step procedure: a first ASP call generates arguments from a given input classical logic knowledge base, while a second ASP call identifies conflicts among the arguments that were created by the first call.

In Assumption-based Argumentation (ABA) structured arguments are created from a rule-based knowledge base [21]. Arguments in this formalism are based on assumptions from which derivations, via the given rules, lead to conclusions of arguments. Two approaches first generate arguments, and conflicts, of a given ABA framework, and then utilize ASP to implement the ABA semantics on the resulting abstract argumentation framework. The first approach [6], implemented in ABAPlus[9], implements ABA+ in this way, which is an extension to ABA that includes preferences among arguments (given as preferences over assumptions). The second approach [43], implemented in aba2af[10], implements ABA semantics. Both approaches do not create all arguments from a given ABA framework, but introduce certain smaller sets of (abstract) arguments that are sufficient for reasoning over the given ABA framework.

Defeasible logic [37] or DeLP programs are as programs in logic programming, yet distinguishing between defeasible and strict rules (also, there is no default negation; on the other hand negation can appear in the head of rules). A dialectical procedure essentially amounting to considering all possible arguments relevant to evaluating a claim is used to determine whether a conclusion is warranted. In [57] the authors study the relationship between DeLP and ASP by defining two possible translations of DeLP programs into ASP programs differing in the treatment of strict rules. Yet in both cases warrant of a literal w.r.t. the DeLP program implies credulous acceptance of the corresponding literal w.r.t. the resulting ASP program.

A more practical use of ASP in the context of DeLP is the work[11] presented in [3]. Here the authors evaluate the use of ASP as an alternative NP-oracle (to SAT) for the two main queries (looking for valid arguments and finding collective conflicts among arguments) underlying the algorithm proposed by the authors to compute the set of warranted and blocked conclusions resulting from a recursive semantics for a generalisation of DeLP: P-DeLP [4]. The latter attaches levels of strength to defeasible rules, formalised as degrees of probabilistic necessity. The recursive semantics [1] on the other hand avoids the potentially exponential number of arguments that need to be considered by the DeLP-style warrant procedure and can be implemented in polynomial space [2].

In [62,63] (revision of [64]) the authors propose a "two-step" approach to instantiation of theories consisting of strict and defeasible (propositional) rules to AFs. The approach in question bypasses the construction of (potentially exponentially many) complex arguments prior to generating an AF as in the more standard approach underlying e.g. deductive argumentation as well as ASPIC+ and discussed in [15]. Instead, the only nodes of the resulting AFs represent

[9] `http://www-abaplus.doc.ic.ac.uk/`
[10] `https://www.cs.helsinki.fi/group/coreo/aba2af/`
[11] `https://github.com/f-guitart/RP-DeLP_solver`

literals and rules of the theories; in fact, the AFs can be computed in polynomial time. In [53] the author reports on ASP encodings for the instantiation of theories via the two-step approach[12]. The encodings are integrated into the DIAMOND system, thus allowing to directly apply this system for reasoning on the instantiated AFs.

ASP encodings are also available for the "direct stable semantics" for knowledge bases consisting of defeasible and strict (first order) rules defined in [56]. The latter is the result of the authors abandoning previous attempts [63,54] at defining a semantics based on instantiation of knowledge bases via abstract argumentation in favor of an approach, where arguments for particular claims are an optional by-product rather than part of the semantics (thus tackling the problems of satisfaction of rationality postulates, over generation of arguments, opacity of attacks, regeneration of the arguments when the knowledge base changes, ...). More specifically, static encodings to disjunctive ASP are provided for existence of a stable model, as well as credulous and skeptical reasoning with respect to propositional defeasible theories[13]; all of these problems are located on the second level of the polynomial hierarchy (the encodings also work for first order defeasible theories, but at the cost of an exponential blowup). There is also ongoing work on making use of the direct stable semantics (and ASP encodings) to provide an argumentation-based interface to a restricted fragment of the controlled natural language ACE [35] extended with constructs for expressing defeasible rules [65,22].

3 Conclusion

In this paper we have surveyed approaches to implement reasoning in formal models of argumentation via ASP. The variety of reasoning tasks, as well as the variety of formal models for which ASP implementations were developed witnesses the wide applicability of both the ASP language and state-of-the-art ASP solvers available today.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF): I2854, P30168-N31, S11409-N23, W1255-N23, and Y698.

References

1. Alsinet, T., Béjar, R., Godo, L.: A characterization of collective conflict for defeasible argumentation. In: Baroni, P., Cerutti, F., Giacomin, M., Simari, G.R. (eds.) Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 216, pp. 27–38. IOS Press (2010)

[12] `https://sourceforge.net/p/diamond-adf/code/ci/master/tree/lib/theorybase.lp`

[13] `https://github.com/hstrass/defeasible-rules`

2. Alsinet, T., Béjar, R., Godo, L., Guitart, F.: Maximal ideal recursive semantics for defeasible argumentation. In: Benferhat, S., Grant, J. (eds.) Proc. SUM. Lecture Notes in Computer Science, vol. 6929, pp. 96–109. Springer (2011)
3. Alsinet, T., Béjar, R., Godo, L., Guitart, F.: Using answer set programming for an scalable implementation of defeasible argumentation. In: Proc. ICTAI. pp. 1016–1021. IEEE Computer Society (2012)
4. Alsinet, T., Chesñevar, C.I., Godo, L., Simari, G.R.: A logic programming framework for possibilistic argumentation: Formalization and logical properties. Fuzzy Sets and Systems **159**(10), 1208–1228 (2008)
5. Atkinson, K., Baroni, P., Giacomin, M., Hunter, A., Prakken, H., Reed, C., Simari, G.R., Thimm, M., Villata, S.: Towards artificial argumentation. AI Magazine **38**(3), 25–36 (2017)
6. Bao, Z., Cyras, K., Toni, F.: ABAplus: Attack reversal in abstract and structured argumentation with preferences. In: An, B., Bazzan, A.L.C., Leite, J., Villata, S., van der Torre, L.W.N. (eds.) Proc. PRIMA. Lecture Notes in Computer Science, vol. 10621, pp. 420–437. Springer (2017)
7. Baroni, P., Caminada, M., Giacomin, M.: Abstract argumentation frameworks and their semantics. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 4, pp. 159–236. College Publications (2018)
8. Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.): Handbook of Formal Argumentation. College Publications (2018)
9. Besnard, P., García, A.J., Hunter, A., Modgil, S., Prakken, H., Simari, G.R., Toni, F.: Introduction to structured argumentation. Argument & Computation **5**(1), 1–4 (2014)
10. Besnard, P., Hunter, A.: Elements of Argumentation. MIT Press (2008)
11. Brewka, G., Diller, M., Heissenberger, G., Linsbichler, T., Woltran, S.: Solving advanced argumentation problems with answer-set programming. In: Singh, S.P., Markovitch, S. (eds.) Proc. AAAI. pp. 1077–1083. AAAI Press (2017)
12. Brewka, G., Ellmauthaler, S., Strass, H., Wallner, J.P., Woltran., S.: Abstract dialectical frameworks. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 5, pp. 237–285. College Publications (2018), also appears in IfCoLog Journal of Logics and their Applications 4(8):2263–2318
13. Brewka, G., Strass, H., Ellmauthaler, S., Wallner, J.P., Woltran, S.: Abstract dialectical frameworks revisited. In: Rossi, F. (ed.) IJCAI. pp. 803–809. IJCAI/AAAI (2013)
14. Brewka, G., Woltran, S.: GRAPPA: A semantical framework for graph-based argument processing. In: Schaub, T., Friedrich, G., O'Sullivan, B. (eds.) Proc. ECAI. Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 153–158. IOS Press (2014)
15. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artif. Intell. **171**(5-6), 286–310 (2007)
16. Caminada, M., Sá, S., Alcântara, J., Dvořák, W.: On the equivalence between logic programming semantics and argumentation semantics. Int. J. Approx. Reasoning **58**, 87–111 (2015)
17. Cayrol, C., Lagasquie-Schiex, M.: Bipolarity in argumentation graphs: Towards a better understanding. Int. J. Approx. Reasoning **54**(7), 876–899 (2013)
18. Cerutti, F., Gaggl, S.A., Thimm, M., Wallner, J.P.: Foundations of implementations for formal argumentation. In: Baroni, P., Gabbay, D., Giacomin, M., van der

Torre, L. (eds.) Handbook of Formal Argumentation, chap. 15, pp. 688–767. College Publications (2018), also appears in IfCoLog Journal of Logics and their Applications 4(8):2623–2706
19. Charwat, G., Dvořák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Methods for solving reasoning problems in abstract argumentation - A survey. Artif. Intell. **220**, 28–63 (2015)
20. Charwat, G., Wallner, J.P., Woltran, S.: Utilizing ASP for Generating and Visualizing Argumentation Frameworks. In: Fink, M., Lierler, Y. (eds.) Proceedings of the 5th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2012). pp. 51–65 (2012)
21. Cyras, K., Fan, X., Schulz, C., Toni, F.: Assumption-based argumentation: Disputes, explanations, preferences. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 7, pp. 365–408. College Publications (2018), also appears in IfCoLog Journal of Logics and their Applications 4(8):2407–2456
22. Diller, M., Wyner, A.Z., Strass, H.: Defeasible acerules: A prototype. In: Gardent, C., Retoré, C. (eds.) Proc. IWCS(1). The Association for Computational Linguistics (2017)
23. Dung, P.M.: 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)
24. Dvořák, W., Dunne, P.E.: Computational problems in formal argumentation and their complexity. In: Baroni, P., Gabbay, D., Giacomin, M., van der Torre, L. (eds.) Handbook of Formal Argumentation, chap. 14, pp. 631–687. College Publications (2018), also appears in IfCoLog Journal of Logics and their Applications 4(8):2557–2622
25. Dvořák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Making use of advances in answer-set programming for abstract argumentation systems. In: Tompits, H., Abreu, S., Oetsch, J., Pührer, J., Seipel, D., Umeda, M., Wolf, A. (eds.) Proceedings of the 19th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2011), Revised Selected Papers. Lecture Notes in Artificial Intelligence, vol. 7773, pp. 114–133. Springer (2013)
26. Dvořák, W., Gaggl, S.A., Linsbichler, T., Wallner, J.P.: Reduction-based approaches to implement Modgil's extended argumentation frameworks. In: Eiter, T., Strass, H., Truszczynski, M., Woltran, S. (eds.) Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 9060, pp. 249–264. Springer (2015)
27. Dvořák, W., Greßler, A., Woltran, S.: Evaluating SETAFs via answer-set programming. In: Thimm, M., Cerutti, F., Vallati, M. (eds.) Proceedings of the Second International Workshop on Systems and Algorithms for Formal Argumentation (SAFA 2018). CEUR Workshop Proceedings, vol. 2171, pp. 10–21. CEUR-WS.org (2018)
28. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument & Computation **1**(2), 147–177 (2010)
29. Eiter, T., Faber, W., Fink, M., Woltran, S.: Complexity results for answer set programming with bounded predicate arities and implications. Ann. Math. Artif. Intell. **51**(2-4), 123–165 (2007)
30. Ellmauthaler, S., Strass, H.: The DIAMOND system for computing with abstract dialectical frameworks. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (eds.)

Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 266, pp. 233–240. IOS Press (2014)
31. Ellmauthaler, S., Strass, H.: DIAMOND 3.0 - A native C++ implementation of DIAMOND. In: Baroni, P., Gordon, T.F., Scheffler, T., Stede, M. (eds.) Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 287, pp. 471–472. IOS Press (2016)
32. Ellmauthaler, S., Strass, H.: go DIAMOND 0.6.6 ICCMA 2017 System Description. `http://argumentationcompetition.org/2017/goDIAMOND.pdf` (2017)
33. Ellmauthaler, S., Wallner, J.P.: Evaluating abstract dialectical frameworks with ASP. In: Verheij, B., Szeider, S., Woltran, S. (eds.) COMMA. Frontiers in Artificial Intelligence and Applications, vol. 245, pp. 505–506. IOS Press (2012)
34. Faber, W., Vallati, M., Cerutti, F., Giacomin, M.: Enumerating preferred extensions using ASP domain heuristics: The ASPrMin solver. In: Modgil, S., Budzynska, K., Lawrence, J. (eds.) Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 305, pp. 459–460. IOS Press (2018)
35. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto controlled english for knowledge representation. In: Baroglio, C., Bonatti, P.A., Maluszynski, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. Lecture Notes in Computer Science, vol. 5224, pp. 104–124. Springer (2008)
36. Gaggl, S.A., Manthey, N., Ronca, A., Wallner, J.P., Woltran, S.: Improved answer-set programming encodings for abstract argumentation. TPLP **15**(4-5), 434–448 (2015)
37. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. TPLP **4**(1-2), 95–138 (2004)
38. Gebser, M., Kaminski, R., Kaufmann, B., Lühne, P., Obermeier, P., Ostrowski, M., Romero, J., Schaub, T., Schellhorn, S., Wanko, P.: The potsdam answer set solving collection 5.0. KI **32**(2-3), 181–182 (2018)
39. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. **187**, 52–89 (2012)
40. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. J. Artif. Intell. Res. **60**, 41–95 (2017)
41. Heissenberger, G., Woltran, S.: GrappaVis - A system for advanced graph-based argumentation. In: Baroni, P., Gordon, T.F., Scheffler, T., Stede, M. (eds.) Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 287, pp. 473–474. IOS Press (2016)
42. Kökciyan, N., Sassoon, I., Young, A.P., Modgil, S., Parsons, S.: Reasoning with metalevel argumentation frameworks in Aspartix. In: Modgil, S., Budzynska, K., Lawrence, J. (eds.) Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 305, pp. 463–464. IOS Press (2018)
43. Lehtonen, T., Wallner, J.P., Järvisalo, M.: From structured to abstract argumentation: Assumption-based acceptance via AF reasoning. In: Antonucci, A., Cholvy, L., Papini, O. (eds.) Proc. ECSQARU. Lecture Notes in Computer Science, vol. 10369, pp. 57–68. Springer (2017)
44. Linsbichler, T., Pührer, J., Strass, H.: A uniform account of realizability in abstract argumentation. In: Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F. (eds.) Proc. ECAI. Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 252–260. IOS Press (2016)
45. Modgil, S.: Reasoning about preferences in argumentation frameworks. Artif. Intell. **173**(9-10), 901–934 (2009)
46. Modgil, S., Prakken, H.: A general account of argumentation with preferences. Artif. Intell. **195**, 361–397 (2013)

47. Nielsen, S.H., Parsons, S.: A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In: Maudet, N., Parsons, S., Rahwan, I. (eds.) Proc. ArgMAS. Lecture Notes in Computer Science, vol. 4766, pp. 54–73. Springer (2006)
48. Nieves, J.C., Cortés, U., Osorio, M.: Preferred extensions as stable models. TPLP **8**(4), 527–543 (2008)
49. Niskanen, A., Wallner, J.P., Järvisalo, M.: Extension enforcement under grounded semantics in abstract argumentation. In: Thielscher, M., Toni, F., Wolter, F. (eds.) Proc. KR. pp. 178–183. AAAI Press (2018)
50. Pührer, J.: ArgueApply: A mobile app for argumentation. In: Balduccini, M., Janhunen, T. (eds.) Proc. LPNMR. Lecture Notes in Computer Science, vol. 10377, pp. 250–262. Springer (2017)
51. Pührer, J.: ArgueApply: Abstract argumentation at your fingertips. KI **32**(2-3), 209–212 (2018)
52. Sakama, C., Rienstra, T.: Representing argumentation frameworks in answer set programming. Fundam. Inform. **155**(3), 261–292 (2017)
53. Strass, H.: Implementing instantiation of knowledge bases in argumentation frameworks. In: Parsons, S., Oren, N., Reed, C., Cerutti, F. (eds.) Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 266, pp. 475–476. IOS Press (2014)
54. Strass, H.: Instantiating rule-based defeasible theories in abstract dialectical frameworks and beyond. J. Log. Comput. **28**(3), 605–627 (2018)
55. Strass, H., Wallner, J.P.: Analyzing the computational complexity of abstract dialectical frameworks via approximation fixpoint theory. Artif. Intell. **226**, 34–74 (2015)
56. Strass, H., Wyner, A.: On automated defeasible reasoning with controlled natural language and argumentation. In: Barták, R., McCluskey, T.L., Pontelli, E. (eds.) Proc. KnowProS. pp. 765–773. AAAI Press (2017)
57. Thimm, M., Kern-Isberner, G.: On the relationship of defeasible argumentation and answer set programming. In: Besnard, P., Doutre, S., Hunter, A. (eds.) Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 172, pp. 393–404. IOS Press (2008)
58. Toni, F., Sergot, M.: Argumentation and answer set programming. In: Balduccini, M., Son, T.C. (eds.) Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday, pp. 164–180. Springer (2011)
59. Wakaki, T., Nitta, K.: Computing argumentation semantics in answer set programming. In: Hattori, H., Kawamura, T., Idé, T., Yokoo, M., Murakami, Y. (eds.) Proc. JSAI and Workshops. Revised Selected Papers. Lecture Notes in Computer Science, vol. 5447, pp. 254–269. Springer (2008)
60. Wakaki, T., Nitta, K., Sawamura, H.: Computing abductive argumentation in answer set programming. In: McBurney, P., Rahwan, I., Parsons, S., Maudet, N. (eds.) Proc. ArgMAS. Revised Selected and Invited Papers. Lecture Notes in Computer Science, vol. 6057, pp. 195–215. Springer (2009)
61. Wallner, J.P.: Structural constraints for dynamic operators in abstract argumentation. In: Modgil, S., Budzynska, K., Lawrence, J. (eds.) Proc. COMMA. Frontiers in Artificial Intelligence and Applications, vol. 305, pp. 73–84. IOS Press (2018)
62. Wyner, A.Z., Bench-Capon, T.J.M., Dunne, P.E.: On the instantiation of knowledge bases in abstract argumentation frameworks. In: Leite, J., Son, T.C., Torroni, P., van der Torre, L., Woltran, S. (eds.) Proc. CLIMA. Lecture Notes in Computer Science, vol. 8143, pp. 34–50. Springer (2013)

63. Wyner, A.Z., Bench-Capon, T.J.M., Dunne, P.E., Cerutti, F.: Senses of 'argument' in instantiated argumentation frameworks. Argument & Computation **6**(1), 50–72 (2015)
64. Wyner, A.Z., Bench-Capon, T.J., Dunne, P.E.: Instantiating knowledge bases in abstract argumentation frameworks. In: Bench-Capon, T., Parsons, S., Prakken, H. (eds.) Proc. AAAI Fall Symposium - The Uses of Computational Argumentation. AAAI Technical Report, vol. FS-09-06. AAAI (2009)
65. Wyner, A.Z., Strass, H.: dARe - using argumentation to explain conclusions from a controlled natural language knowledge base. In: Benferhat, S., Tabia, K., Ali, M. (eds.) Proc. IEA/AIE. Lecture Notes in Computer Science, vol. 10351, pp. 328–338. Springer (2017)