# Towards Argument Representational Tools for Hybrid Argumentation Systems

María Paula González[1,2], Sebastian Gottifredi[1,2], Alejandro J. García[1,2], and Guillermo R. Simari[2]

[1] National Council of Scientific and Technical Reseach CONICET, Argentina
[2] Computer Science Department, Universidad Nacional del Sur
Av Alem 1253 – 8000 Bahía Blanca, Argentina
{mpg,sg,ajg,grs}@cs.uns.edu.ar

**Abstract.** Argumentation Systems are reasoning systems that provide automatic computation of arguments. "Argument Assistant Systems" are graphic-oriented tools for supporting end-users to manipulate arguments. Recently, the novel family of "Hybrid Argumentation Systems" (HAS) has emerged, combining these two approaches. Even when some HAS have been presented, either they show in the interface only final results of the computation of the dispute situation under consideration, or have not explicit considered usability features focused on real final users. Besides, current semantic goes from the definition of theoretical considerations to the graphical representation of the dispute situation under consideration, avoiding the direct manipulation of arguments is a graphical fashion. This paper discusses lessons learned at the development of DeLP Client, a particular HAS software oriented towards end-users where main goals include going beyond the above limitations. To achieve usability goals, some usability-oriented design guidelines recently proposed for the argumentation systems domain are considered.

**Keywords:** Knowledge Representation, Defeasible Argumentation, Hybrid Argumentation Systems, Usability Guideline.

## 1 Introduction and Motivation

Argumentation is an important aspect of human decision making. In any argumentation process first a construction of arguments supporting and against a statement is made, second the set of warrant (acceptable) arguments is defined and finally it is decided whether the statement can be ultimately accepted or not. Over the past decade, theoretical advances in the area have consolidated different computational models, going from pure Argumentation Systems (AS) [1] that provides automatic calculus of arguments to more user-oriented tools called Argument Assistant Systems (AAS) [2] [3], where the goal is to assist the user in the process of arguing, rather than to provide complex reasoning tasks.

In between, a novel family of argumentation systems called Hybrid Argumentation Systems (HAS) has emerged, combining the above two approaches. User oriented tools which provide an aid for drafting and generating arguments are complemented

G. Salvendy, M.J. Smith (Eds.): Human Interface, Part II, HCII 2011, LNCS 6772, pp. 236–245, 2011.

with capabilities for automatic calculus of arguments that help to come to a decision when complex scenarios of arguments have to be considered. HAS general abstract frameworks have to be distinguished from the more concrete models that we have called "concrete HAS" ($HAS_C$), where a particular logic language is involved. Even when different HAS have been presented in the last years, they suffer a number of limitations: either they show in the interface only final results of the computation of the dispute situation under consideration instead of depicting the most significant intermediate steps that leads to give some argument status (e.g. warrant, undecided, etc.); or are not intended for final users. Besides, in most cases usability is not explicitly considered. However, facing end users with real interactive interfaces is essential to ensure HAS penetration beyond academia. In addition, achieving an appropriate degree of quality (in particular associated with usability) is crucial to promote good practices and acceptance of this kind of tools. In that respect, note that recently a set of usability-oriented guidelines have been proposed for the development of AAS [4], advancing towards an standardization about the way arguments can be sensibly and clearly presented to users, especially when they are defeasible.

This paper discusses lessons learned at the development of the concrete $HAS_C$ DeLP Client oriented towards end-users. The tool is based on Defeasible Logic Programing DeLP [5], which has been successfully embedded in real-world applications (e.g. recommender systems [6], decision support systems [7], and CSCW [8]). On the basis of the above usability-oriented guidelines, a full implemented prototype is presented. A preliminary usability inspection is sketched. Our final goal focuses on the design, implementation and evaluation of qualify HAS software tools for creating, drafting, calculating, and analyzing arguments.

## 2 Characterizing Hybrid Argumentation Systems

Argumentation Systems (AS) are increasingly being considered for applications, constituting an important component of multi-agent systems for negotiation, problem solving, and for the fusion of data and knowledge [9]. In this context, Hybrid Argumentation Systems (HAS) combine the power of the AS with some user-oriented facilities inherited from Argument Assistant Systems (AAS) [10] [11] [12] [13]. As pointed out in [2], AAS provide often a realization of a formal argumentation theory, offering a good test bed for analyzing the advantages and disadvantages of the actual application of the theory. They have to be distinguished from AS automated reasoning systems; the latter can do complex reasoning tasks for the user, whereas AAS's goal is not to replace the user's reasoning, but rather to assist him in this process.

The term HAS was coined by Hunter at [14], where an early discussion about the necessity of combining "formal" and "informal" argumentation models (AS and AAS respectively) was presented, and a first attempt to characterize common HAS features was shown. Going beyond [14], alternative modelling of argument leads to the distinction between what we have called "concrete HAS models" ($HAS_C$) -where a specific logical language underlays the definition of arguments and the notion of attack- and general HAS. Examples of $HAS_C$ systems include [15] [16] [17] [18] and [19], among others. On the other hand, general HAS (as [20], [21] or [22]) have an abstract structure, including generic argument representation and a binary relation

between them called "attack relation". As posted in [23], abstracting away from the structure and meaning of arguments and attacks enables the study of properties which are independent of any specific aspect, but limiting expressiveness and applicability.

Despite of their differences, all HAS systems include an embedded AS that allows to determine when a given argument can be considered as ultimately acceptable with respect to the available knowledge by means of some recursive analysis, which takes the form of a tree-like structure called dialectical tree in the particular case of defeasible logic-based $HAS_C$ as [5] or [24]. Intuitively, in a dispute scenario conflicting arguments may emerge: an argument A attacks another argument B whenever both of them cannot be accepted at the same time, as that would lead to contradictory conclusions. The notion of defeat comes then into play to decide which argument should be preferred. An argument A defeats an argument B whenever A attacks B, and besides, A is preferred over the attacked part in B (with respect to some preference criterion). The criterion for defeat can be defined in many ways. As a generic criterion, it is also common to prefer those arguments which are more direct or more informed. This is known as the specificity principle [1]. The notion of defeat among arguments may lead to complex "cascade" situations: an argument A may be defeated by an argument B, which in turn may be defeated by an argument C, and so on. Besides, every argument may have on its turn more than one defeater. Arguments must additionally satisfy the requirement of consistency (not include contradictory propositions) and minimality (not including repeated or unnecessary information).

In addition, HAS should provide all the facilities included in AAS for drafting and generating arguments, assisting the user in his reasoning process. This assistance involves several aspects of the argumentation process, e.g. keeping track of the issues that have been raised, assumptions that have been made, evaluating the justification status of the statements involved in the argumentation process, etc. Indeed, some features have to be included in the HAS interfaces, as they constitute common elements at AAS [4] [14]. First, they convey the representation of some user mental model, (i.e., all the cultural and personal-biased users' perceptions and assumptions, as well as their pre-conceptions about how the tasks performed by the system are solved in real world and consequently how the system is expected to react), together with the interaction style (including both physical and mental actions). Additionally, feedback and support is usually included (explicit current system status; prevention and recovering from errors and misuse, e.g. by means of help and documentation, undo options, etc.) as well as diverse interoperability facilities (as links to multimedia elements). Besides, there are some common features in AAS interfaces typically associated with the argumentation process itself. Three central features are the visual argument representation (including the recognition of different types of arguments, their statuses, etc.), the modeling of conflict among arguments which allows the user to recognize the argumentation situation under consideration, and the preference criteria associated with the possibility of visualizing or deducing how the conflict among arguments is resolved.

As mentioned above, some implemented HAS were designed to show in the interface only final results of the computation of the dispute situation under consideration. Others are intended for agents rather than for final users [19]. Besides, in most cases usability is not explicitly considered. Note that these limitations do not invalidate the qualitative advances conquered in the last years. However, from the

end-user perspective visualizing a comprehensible and graphic oriented representation of the intermediate steps to determine the status of an argument (e.g. warrant, undecided, etc.) enhance the understanding of final results, giving more adequate support and feedback. Besides, note that the user's acceptance regarding HAS will be directly proportional to its interface quality in use. As we will describe in the next section, to cope with the above situation recently a novel $HAS_C$ system is being developed, including the AS DeLP and a user-oriented interface. Some usability-oriented guidelines focused on AAS.

## 3 The Proposal

As stated in [11], a significant strand in AS research focuses nowadays on the design, implementation and evaluation of practical software tools for creating and analyzing arguments. Consequently, we started comparing the existing AS and AAS to characterize common features in their interfaces, plus the minimal set of requirement for designing and developing usable HAS. Then, based on DeLP [5] and a previous visualization tool developed for DeLP answers [25], we developed the $HAS_C$ prototype DeLP Client including the usability-oriented guidelines shown at [4].

### 3.1 Designing and Developing Usable $HAS_C$ Systems

Developing high quality $HAS_C$ systems is challenge. Over the last 10 years, the recommendations early proposed at [14] still remind valid. However, a more complex scenario justifies a revision of these pioneering ideas. First, a minimal set of operations has to be covered. Taking as starting point [3] and [14], all the HAS features described in Section 2 have to be covered (e.g. the user mental model, a visual argument representation, etc.). In addition, some requirements to ensure the visualization of the automatic calculus of arguments have to be presented, including the representation of the intermediate steps that lead to particular conclusions (argument status). Second, note that the quality of the HAS interfaces will play a key role respecting the user experience. In particular, usability - formally defined by ISO 9241-11 as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”- will play a major role. In that respect, our learned lessons include detecting the necessity of standardization for HAS respecting usability. As a first step, a minimal set of usability-oriented guidelines has been proposed for AAS [4], where each guideline is instantiated by means of questions and recommendations, and different usability principles were identified to evaluate guidelines quality in use. Third, current HAS implementations provide a one-way interactive style, in the sense that the graphical representation is static, just mirroring the output of the automatic computations. On the contrary, some AAS as Araucaria [10] or Compendium [12] are flexible enough to admit manipulation of arguments in a direct fashion. Indeed, it is possible to move arguments, resize them, link them between each other (e.g. attack or contra-attack, support, add more specificity, etc.). Consequently, it would be desirable to define a novel double-way interactive style for HAS to include direct manipulation of arguments. In the case of the $HAS_C$ the challenge will be to link user actions at the

system interface with the dynamic modification of the dialectical trees that compute the current status of all arguments under consideration. In addition, if the direct manipulation includes the possibility of "drag and drop" information for external sources (as in [10]), then the knowledge base should be updated.

### 3.2 The $HAS_C$ DeLP Client: A Prototype

As stated above, our proposal includes the general-purpose AS Defeasible Logic Programming (DeLP) [5], which uses a representational language for writing sentences that follows Prolog syntaxs. In DeLP two kinds of knowledge are included. Strict knowledge or $K_S$ (denoted Head <- body) corresponds to the knowledge which is certain, as statements or undisputable facts about the world, or mathematical truths (e.g. implications of the form $(\forall(x)\ P(x)\rightarrow Q(x))$. The strict knowledge is consistent, i.e. no contradictory conclusions can be derived from it. On the other hand, defeasible knowledge or $K_D$ (denoted head -< body) corresponds to that knowledge which is tentative, modelled through "rules with exceptions" (defeasible rules) of the form "if P then usually Q" (e.g., "if something is a bird, it usually flies"). Such rules model our incomplete knowledge about the world, as they can have exceptions (e.g., a penguin, a dead bird, etc.). An argument A for a claim c is basically some "tentative proof" or proof tree (formally, a ground instance of a subset of $K_D$) for concluding c from $A \cup K_S$. Figures 1 and 2 (left) show DeLP syntax, including $K_S$ and $K_D$ examples.

Given an argument A1 for some conclusion C, DeLP builds all the sequences of arguments $[A_1, A_2, \ldots, A_k]$ such that every argument $A_i$ (except the first) defeats the previous argument in the sequence. These sequences are known as argumentation lines (or dialogue line) and model dispute dialogues between two parts called Proponent and Opponent. An argument A considered as finally acceptable (warranted or undefeated) if every argumentation line starting in A has an odd number of arguments. This accounts to say that A has "survived" all possible attacks (i.e., every attack to A is successfully defended in every argumentation line). Typically, when there are several argumentation lines, they are represented in a tree structure (dialectical tree) which shows all possible argumentation line that start by a given argument. Queries are solved by computing these dialectical trees, answering "yes" (only if there is a warranted argument supporting the query), "no" (only if there is a warranted argument supporting the contrary of the query.), or "undecided if we are in a situation in which we cannot decide (none of the above cases hold).

Figures 1 and 2 show different screenshots of the DeLP Client $HAS_C$ prototype.[1] While Figure 1 includes the toy "Bird" example related to the information "if something is a bird, it usually flies", Figure 2 is associated with a more complex example called "Many Trees" where the value of visualizing intermediate steps when many arguments are jointly considered is appreciated. The AAS part (both figures, left) includes two main horizontal panels. On the top, a Query Manager where queries can be posted by the user; on the bottom the editable DeLP Program Manager to write statements (or open an existing file) that describes the current dispute situation under consideration, including undo options (e.g. clear button) and the facility to load the statements at the DeLP engine, the embedded AS that was described above.

[1] Current version deployed during 2010 at LIDIA Lab (http://lidia.cs.uns.edu.ar/).

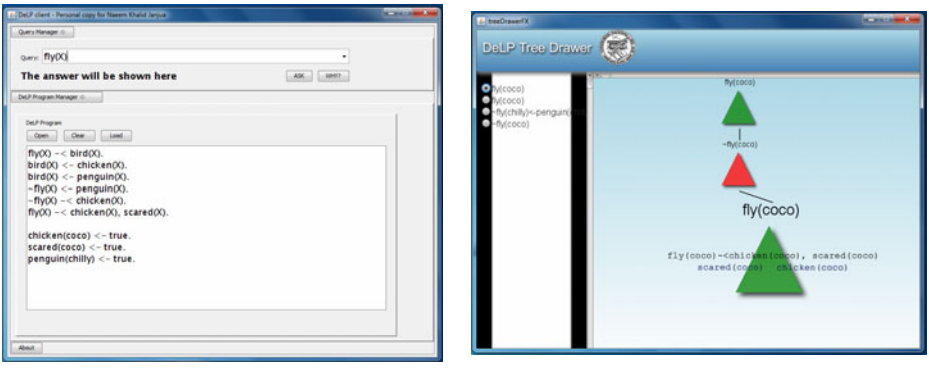


**Fig. 1.** Client Screenshots for the "Bird" example. AAS part (left). DeLP Tree Drawer showing intermediate steps for the claim "fly(coco)?" with zoom at the bottom argument. Top and bottom triangle are green, intermediate triangle is red (right).

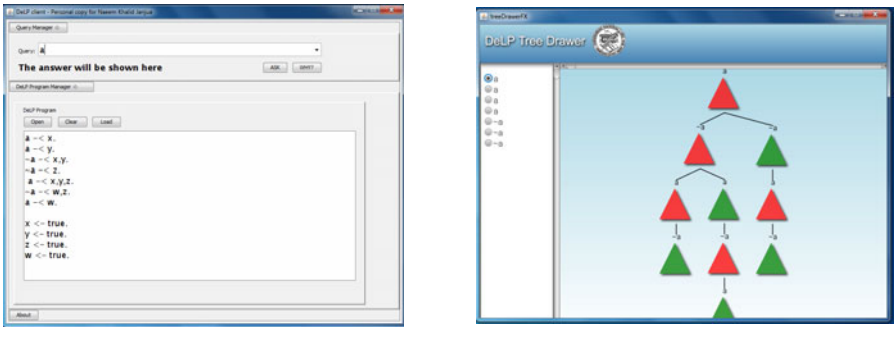


**Fig. 2.** DeLP Client Screenshots for the "Many Trees" example. AAS part (left). DeLP Tree Drawer showing evidence for the claim "a?" (right).

Once the sentences are introduced, users can post queries at the Query Manager to ask for any claim; and rely on DeLP to compute the status of the involved arguments. To respond a query, the DeLP Client opens the DeLP Tree Drawer (Figures 1 and 2, left), where two panels show the final argument status plus the most significant intermediate steps that lead to them. Dialectical trees with triangular nodes and arrows model the arguing situation: each triangle represents an argument (triangles are a common drawing for proof trees), and arrows indicate relationships and the inference direction. The left panel shows all the possible dialogue lines that can be computed associated with the current query, each of them identified by his root. The best explanation is detailed in the right panel, where green nodes depict undefeated arguments and red nodes the defeated ones. As in [25], if any node is clicked, the dialectical tree expands itself detailing the internal structure of the argument. Besides, both the statements and the related trees can be exported.

## 4 Usability Inspection: Discussion

Once the DeLP Client prototype was deployed, a preliminary usability inspection was carried outs to assess the real scope of the main interface features (see Section 2) that account for the usability-oriented guidelines (UG) underlying the design. The inspection was performed at the LIDIA Labs by experts in usability experienced on the AS area. Four different computers were used to simulate user's scenarios. Further usability studies involving real users are needed to validate the conclusions discussed here. The interface seems easy to learn. Respecting the user mental model (UG UM#1 to UM#3), the DeLP Client matches both underlying theory and real world, since the fundamental elements of the theory (arguments, attacks, dialectical trees, automatic calculus of arguments, etc.) were represented. Accepted levels of predictability seems to be achieved, as the node-oriented graphical representation of AAS was followed (covering UG UM#3), avoiding the inclusion of Prolog-like syntactic. The domain is explicit enough, and no other facilities beyond the ones associated with arguing were included. Maybe the position of the Program Manager and Query Manager panels should be interchanged to show at the top the Program Manager, as in real world at least a sentence should be posted before arguing. Note that the top panel is not accessible before introducing information in the bottom one.

The interaction style (UG IS#1 to IS#7) sketches a future WIMP (Windows, Icons, Menus, Pointers) fashion. It combines form filling for the input of sentences with a still simple menu selection (currently the quantity of available options do not deserve pull-down menus, and options are displayed with boxes in horizontal lines). Main options are activated by means of mouse clicks and, as discussed before, direct manipulation of arguments associated with a novel "double way" interactive style has been characterized but still reminds undeveloped. Covering UG IS#1, when the DeLP FX Visualizer is activated the user control is slightly reduced, showing coherence with the underlying theory (the automatic calculation of arguments at $HAS_C$). Besides, a minimalist style was adopted (UG IS#3), even when the absence of all the tool final facilities makes difficult to ensure that the current information architecture (simple and clear) will be preserved and not necessary interaction styles will be included (UG IS#5). Maybe the Prolog-like syntax at the Program Manager should be replaced (or complemented) with a more user-oriented language to better minimize technical considerations (UG IS#4). The prototype does not provide configurable issues (UG IS#7). However, respecting the UG IS#7 different elements can be easily included in a single view: e.g., statements can be added to the bottom of the DeLP FX Visualizer to reinforce the user perception of the discussion under consideration.

The visual argument representation plus the conflict among argument modelling (UG VR#1 to VR#3 and CA#1 to CA#3) are remarkable. Indeed, main goals of the proposal include visualizing a comprehensible and graphic oriented representation of the intermediate steps that leads to give some argument status. In that respect, the double panel screen of the DeLP FX Visualizer enrich user experience by providing a comprehensible and intuitive representation of both the final result and the alternative answers to a particular query. In particular, the expert opinions suggested that arguments are clearly represented by the tree nodes (UG VR#1). The selection of red and green colors to distinguish final status of arguments helps to achieve affordance and predictability of guideline VR#2, as red is usually associated with "no" symbols

(not supported arguments) and green is the opposite (as in the semaphores). Besides, the arguments relevance (UG VR#3) is visualized in an easy to learn and consistent way, since tree-like structures position the argument under consideration at the root and then alternate the attacks and supports going down, following prior user knowledge respecting trees hierarchy. Alternative modelling of the arguing situation currently under consideration is not provided (UG CA#2). Different views to allow zoom in and out are nicely included at the DeLP FX Visualizer (UG CA#3); in a flexible, easy and consistent way. However, the prototype under evaluation can't compete with the flexibility provided by some AAS interfaces (as in [12]). Respecting the preference criteria (UG PC#1 and PC#2), the usability inspection could not arises any conclusion. Indeed, real final user opinions are irreplaceable to ponder the achievement of these guidelines. Interoperability related guidelines (UG IO#1 and IO#2) are still inefficient, as only a set of pre-written sentences saved as a .delp file can be imported. However, note that other $HAS_C$ also do not cover these guidelines, leading to the definition of an Argument Interchange Format (AIF) still under discussion. On the contrary, guidelines regarding feedback and support (UG FS#1 to FS#5) are reasonable accomplished. Indeed, undo options are included, and error prevention achieved by means of the described interaction style. Finally, the guidelines associated with collaboration were omitted as the current prototype is designed for individual users.

## 5 Related Work

To the best of our knowledge, there is no another approach similar to the one discussed here. A relevant implemented $HAS_C$ is the agent-oriented MARGO [19], devoted to practical reasoning about service composition. MARGO embeds the CaSAPI system [24], an AS general-purpose engine for assumption-based argumentation implemented in SICStus Prolog. MARGO differs from our approach not only in the underlying logic structure of arguments ([5] versus assumption-based argumentation) but also in the information visualization. While we provide a node-oriented graphical representation, MARGO depicts a Prolog-like syntactic, with statements listed as in command-line interfaces. An example of a $HAS_C$ that has still not includes a final user interface is the [26] system.

Recently some interesting HAS abstract frameworks have been implemented as real world applications. An example is the Dungine Java Reasoner by South et al. [21], successfully integrated with Araucaria [10] using the ArgKit library.[2] Another relevant approach is presented at [munoz09], where the authors propose the materialization of a complete argumentation system ready to be built in conventional agent software platforms. In a more general perspective, some multi-semantic argumentation engines have also been developed with the capability to refer many semantics in the same module at the same time. Interesting examples are the MoDiSo environment,[3] and the web-based LASAD system[4]. Finally, note that the term Hybrid Argumentation was used in [17], but with a different meaning that the one introduced

[2] http://www.argkit.org
[3] http://www.cs.ait.ac.th/~dung/modiso/About.html
[4] http://cscwlab.in.tu-clausthal.de/lasad/

here. While our goal is to characterize the particular family of HAS, [17] refers to a concrete computational model for a form of argumentation that is a hybrid between abstract and assumption-based argumentation.

## 6 Conclusion

Hybrid Argumentation Systems (HAS) emerged in the last years as a natural way of coping with knowledge management and decision making in dispute scenarios where incomplete or contradictory information has to be handle. They offer an interface including user-oriented facilities for visualizing, creating, drafting, and analyzing arguments derived from Argument Assistance Systems [2] [Verjeij 2007], plus an underlying pure Argumentation System (AS) [1] for calculating attack and other relationships between them. This paper discusses the lessons learned at the development process of an $HAS_C$ aimed to mirror not only final results of the computation but also most significant intermediate steps that leads to give some argument status (e.g. warrant, undecided, etc.). Main contributions are related to the visualization of the intermediate steps that support argument calculation in a natural and intuitive way, as well as the consideration of the usability-oriented guidelines shown at [4]. The AS DeLP [5] underlies the proposal.

Future work includes the performance of alternative full usability evaluation including real final users. On the basis of the obtained results, an incremental iterative development process based on a Usability Engineering approach has to be carried out over current implementation. At near future cycles of that process direct manipulation of arguments has to be considered, leading to a revision of the questions associated with every usability-design guideline at [4] to cover it.

**Acknowledgments.** This paper was funded by Projects PIP-CONICET 112-200801-02798 and UNS PGI 24/ZN18 (Argentina); and TIN2008-06596-C02-01 (Spain).

## References

1. Chesñevar, C.I., Maguitman, A., Loui, R.: Logical Models of Argument. ACM Computing Surveys 32(4), 337–383 (2000)
2. Verheij, B.: Artificial argument assistants for defeasible argumentation. Journal of Artificial Intelligence 150(1-2), 291–324 (2003)
3. Verheij, B.: Argumentation Support Software: Boxes-and-Arrows and Beyond. Law, Probability & Risk 6, 187–208 (2007)
4. González, M.P., Chesñevar, C., Pinkwart, N., Gomez Lucero, M.: Developing Argument Assistant System from a Usability viewpoint. In: Proc. KMIS 2010, pp. 157–163 (2010)
5. García, A., Simari, G.: Defeasible Logic Programming: An Argumentative Approach. Theory and Practice of Logic Programming 4(1), 95–138 (2004)
6. Brena, R., Chesñevar, C.: Information Distribution Decisions Supported by Argumentation. In: Encyclopaedia of Decision Making and Decision Support Technology, Information Science Reference, vol. II, pp. 489–495 (2008), ISBN 978-1-59904-843-7
7. Williams, M., Hunter, H.: Harnessing ontologies for argument-based decision-making in breast cancer. In: Proceedings of ICTAI 2007, pp. 254–261 (2007)

8. González, M.P., Penichet, V.M.R., Simari, G.R., Tesoriero, R.: Development of CSCW interfaces from a user-centered viewpoint: Extending the TOUCHE process model through defeasible argumentation. In: Kurosu, M. (ed.) HCD 2009. LNCS, vol. 5619, pp. 955–964. Springer, Heidelberg (2009)
9. Bench-Capon, T., Dunne, E.: Argumentation in Artificial Intelligence. Int. Journal on Artificial Intelligence 171, 619–641 (2007)
10. Reed, C., Rowe, G.: Araucaria: Software for Argument Analysis, Diagramming and Representation. Int. Journal on Artificial Intelligence Tools 14, 961–980 (2004)
11. Buckingham Shum, S.: Cohere: Towards Web 2.0 Argumentation. In: Proc. Int. Conf. COMMA 2008, pp. 97–108. IOS Press, Amsterdam (2008)
12. Okada, A., Buckingham Shum, S., Sherborne, T. (eds.): Knowledge Cartography: Software Tools and Mapping Techniques. Advanced Information and Knowledge Processing Series. Springer, Heidelberg (2008), ISBN 978-1-84800-148-0
13. Van den Braak, S., Vreeswijk, G., Prakken, H.: AVERs: an argument visualization tool for representing stories about evidence. In: Proc. of the 11th ICAIL, pp. 11–15 (2007)
14. Hunter, T.: Hybrid argumentation systems for structured news reports. The Knowledge Engineering Review 16(4), 295–329 (2001), ISSN 0269-8889
15. Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. In: Artificial Intelligence, vol. 128, pp. 203–235 (2001)
16. Rahwan, I., Amgoud, L.: An Argumentation-based Approach for Practical Reasoning. In: 5th AAMAS 2006, pp. 347–354. ACM Press, New York (2006)
17. Gaertner, D., Toni, F.: Hybrid argumentation and its properties. In: Proc. 2nd Int. Conf. COMMA, pp. 183–195. IOS Press, Amsterdam (2008), ISBN: 978-1-58603-859-5
18. Kakas, A.C., Toni, F.: Computing Argumentation in Logic Programming. Journal of Logic and Computation 9, 515–562 (1999)
19. Morge, M.: The hedgehog and the fox. In: Rahwan, I., Parsons, S., Reed, C. (eds.) Argumentation in Multi-Agent Systems. LNCS (LNAI), vol. 4946, pp. 114–131. Springer, Heidelberg (2008)
20. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument and Computation 1, 93–124 (2010)
21. South, M., Vreeswijk, G., Fox, J.: A Java Dung Reasoner. In: Proc. Int. Conf. COMMA 2008, pp. 360–368. IOS Press, Amsterdam (2008), ISBN: 978-1-58603-859-5
22. Brewka, G., Gordin, T.: Carneades and Abstract Dialectical Frameworks: A Reconstruction. In: Proc. COMMA 2010, pp. 3–12. IOS Press, Amsterdam (2010), ISBN 978-1-60750-618-8
23. Baroni, P., Giacomin, M.: Semantics of Abstract Argument Systems. In: Argumentation in Artificial Intelligence, pp. 25–44. Springer, Heidelberg (2009), ISBN: 978-0-387-98196-3
24. Gaertner, D., Toni, F.: Computing Arguments and Attacks in Assumption-Based Argumentation, vol. 22 (6), pp. 24–33. IS, IEEE Computer Society, Los Alamitos (2007), ISSN 1541-1672
25. Escarza, S., Castro, S., Martig, S.: DeLP Viewer: a Defeasible Programming Visualization Tool. In: PROC. XVCACIC, pp. 556–565 (2009), ISBN: 978-897-24068-4-1
26. Williams, M., Hunter, H.: Harnessing ontologies for argument-based decision-making in breast cancer. In: Proceedings of ICTAI 2007, pp. 254–261 (2007)