WizArg: Visual Argumentation Framework Solving Wizard

Ignasi GÓMEZ-SEBASTIÀ, Juan Carlos NIEVES [a,1]
[a] *Knowledge Engineering and Machine Learning Group*

Abstract. Extension-based argumentation semantics have shown to be a suitable approach for performing practical reasoning. An important concern in extension-based-argumentation semantics is the *computational complexity* of the decision problems that has been shown to range from NP-complete to $\Pi_2^{(p)}$-complete.

In this paper, we introduce a generic extension-based argumentation semantics solver, that is called WizArg. The WizArg project consists in two main components: The WizArg Argumentation library (that can be used by any generic software to solve argumentation frameworks and answer questions about extension-based-argumentation semantics) and the WizArg front-end (a generic software component that allows users to create, save, and load their own argumentation frameworks).

Keywords. Argumentation Theory, Argumentation-reasoning tool, Answer-set programming.

Introduction

Argumentation theory is a formal discipline within Artificial Intelligence where the aim is to make a computer assist in or perform the act of argumentation. During the last years, argumentation has been gaining increasing importance in Multi-Agent Systems (MAS), mainly as a vehicle for facilitating *rational interaction* (i.e. interaction which involves the giving and receiving of reasons) [2]. An agent may also use argumentation techniques to perform its individual reasoning on the need of making decisions under complex preference policies or in highly dynamic environments.

Although several approaches have been proposed for capturing representative patterns of inference in argumentation theory, Dung's approach, presented in [4], is a unifying framework which has played an influential role on argumentation research and Artificial Intelligence (AI). Dung's approach is regarded as an abstract model where the main concern is to find the set of arguments which are considered as acceptable *i.e.*, to find sets of arguments which represent coherent points of view. The strategy for analyzing the attack relationship, and then inferring the sets of acceptable arguments, is based on extension based semantics such as *grounded*, *stable*, *preferred* and *complete* semantics. Even though each of these argumentation semantics represents different patterns of selection of arguments all of them are based on the concept of an *admissible set*. Some authors have pointed out that the grounded, stable and preferred semantics are of particular

[1]Departament de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya (UPC) C/Jordi Girona 1-3, E-08034 , Barcelona, Spain. E-Mail:**{igomez, jcnieves}@lsi.upc.edu**

	Instance	Decision Question	Complexity
(a)	$AF = \langle AR, attacks \rangle, x \in AR$	Is x in any *preferred* extension?	NP-complete
(b)	$AF = \langle AR, attacks \rangle, x \in AR$	Is x in any *stable* extension?	NP-complete
(c)	$AF = \langle AR, attacks \rangle$	Does AF have a *non-empty* preferred extension?	NP-complete
(d)	$AF = \langle AR, attacks \rangle$	Does AF have any stable extension?	NP-complete
(e)	$AF = \langle AR, attacks \rangle, x \in AR$	Is x in *every* stable extension?	CO-NP-Complete
(f)	$AF = \langle AR, attacks \rangle, x \in AR$	Is x in *every* preferred extension?	$\Pi_2^{(p)}$-complete

Figure 1. Decision problems in finite argumentation frameworks

interest since they capture representative patterns of inference in argumentation theory [1,2].

Even thought Dung's approach is a versatile and powerful tool for the abstract analysis of defeasible reasoning, one can find some potential problems, *i.e.*, emptiness, non-existence, multiplicity [2]. In order to manage these problems, one can find different approaches for defining new extension-based argumentation semantics such as the based on strongly connected components [1] and logic programming semantics [10]. Among these new argumentation semantics, the CF2 argumentation semantics is a representative semantics which overcomes some problems of extension-based semantics based on admissible sets. CF2 was introduced in [1] and has been characterized in terms of logic programming semantics [10,11].

Another important concern in abstract argumentation theory is the *computational complexity* of the decision problems that has been shown to range from NP-complete to $\Pi_2^{(p)}$-complete. A summary of this is given in Figure 1 (taken from [5]). Observe that the figure only presents complexity results with respect to the stable and preferred semantics; however, one can see that the CF2 semantics are, in the best case, as complex (in computational terms) as the preferred semantics.

From Figure 1, one can see that the computational complexity of the decision problem of argumentation semantics is hard. However, recognizing the benefits of Dung's approach, a number of algorithms has been proposed in the literature [9,11]. There are several *platforms/tools* for implementing argumentation-based systems, *e.g.*, [6]. However, they do not consider extension-based argumentation semantics, *e.g.*, preferred semantics, CF2 semantics. The lack of libraries supporting extension-based argumentation inference increase the gap between theoretical argumentation results and extension-based argumentation systems.

Nowadays in the close relationship between extension-based argumentation semantics and logic programming semantics, there are novel results which have shown that extension-based argumentation semantics can be characterized in terms of logic programming semantics [9,11]. In fact, not only a logic programming approach can characterize extension-based argumentation semantics but also it can define new extension based argumentation semantics [10]. These results suggest some approaches for using algorithms of general purpose such as the algorithm of the answer set solver DLV [3] in order to implement argumentation inference.

In this paper we introduce the *WizArg*[2] project, which aims to provide a library that enables generic computational processes (e.g. Java processes) to use argumentation metainterpreters. The project includes a demonstration of libraries usage in the form of

[2]http://sourceforge.net/projects/wizarg/

a front-end where users can design their own argumentation frameworks, solve them using the WizArg libraries and visualize the results (that is, the classification of each argument as justified, defeated or irresolvable). At the same time, this paper shows how a generic-purpose ASP (Answer Set Programming) algorithms can be used to explore argumentation-based reasoning methods. The current version of WizArg gives support to the grounded, stable, preferred and CF2 argumentation semantics in order to give answer to questions of the form presented in Figure 1.

The rest of the paper is structured as follows: In §1, some concepts of argumentation theory are presented. In §2, the WizArg library is described. In the last section, our conclusions and the issues of our future work are outlined.

1. Background

This section is going to present the formal definitions of the extension-based semantics which are supported by the *WizArg* library, *i.e.*, the grounded, stable, preferred and CF2 semantics, and a short overview of an argumentation labeling process.

1.1. Extension based argumentation semantics based on admissible sets

A fundamental Dung's definition is the concept called argumentation framework which is defined as follows:

Definition 1 *[4] An argumentation framework is a pair* $AF = \langle AR, attacks \rangle$*, where* AR *is a set of arguments, and* attacks *is a binary relation on* AR*,* i.e.attacks $\subseteq AR \times AR$*.*

Following Dung's reading, we say that A attacks B (or B is attacked by A) if $attacks(A, B)$ holds. Similarly, we say that a set S of arguments attacks B (or B is attacked by S) if B is attacked by an argument in S.

Definition 2 *[4]*

- *A set S of arguments is said to be conflict-free if there are no arguments A, B in S such that A attacks B.*
- *An argument* $A \in AR$ *is acceptable with respect to a set S of arguments if and only if for each argument* $B \in AR$*: If B attacks A then B is attacked by S*
- *A conflict-free set of arguments S is admissible if and only if each argument in S is acceptable w.r.t. S.*

The (credulous) semantics of an argumentation framework can be defined by the notion of the *stable and preferred extensions*.

Definition 3 *[4]*

- *A preferred extension of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of AF.*
- *A conflict-free set of arguments S is called a stable extension if and only if S attacks each argument which does not belong to S.*

Dung also defined a skeptical semantics which is called grounded semantics and it is defined in terms of a *characteristic function*.

Definition 4 *[4] The characteristic function, denoted by F_{AF}, of an argumentation framework $AF = \langle AR, attacks \rangle$ is defined as follows:*
$F_{AF} : 2^{AR} \rightarrow 2^{AR}$
$F_{AF}(S) = \{A |$ *A is acceptable w.r.t. S* $\}$

Definition 5 *[4] The grounded extension of an argumentation framework AF, denoted by GE_{AF}, is the least fixed point of F_{AF}*

1.2. Semantics $CF2$

We present some definitions *w.r.t.* the argumentation semantics CF2. The details of these definitions are presented in [1].

Definition 6 *[1] Given an argumentation framework $AF = \langle AR,$* attacks$\rangle$*, the binary relation of* path-equivalence *between nodes, denoted as $PE_{AF} \subseteq (AR \times AR)$, is defined as follows:*
— $\forall a \in AR$, $(a, a) \in PE_{AF}$,
— given two distinct nodes a, $b \in AR$, $(a, b) \in PE_{AF}$ if and only if there is a path from a to b and a path from b to a.

Given an argumentation framework $AF = \langle AR, attacks \rangle$, the *strongly connected components* of AF are the equivalent classes of nodes which are defined according to path-equivalence relaction. The set of the strongly connected components of AF is denoted as $SCCS_{AF}$. Given a node $a \in AR$, the strongly connected component a belongs to is denoted as $SCC_{AF}(a)$.

Definition 7 *[1] Let $AF = \langle AR,$* attacks$\rangle$ *be an argumentation framework, and let $S \subseteq AR$ be a set of arguments. The* restriction *of AF to S is the argumentation framework $AF \downarrow_S = \langle S,$* attacks $\cap (S \times S)\rangle$.

Considering an argumentation framework, $AF = \langle AR, attacks \rangle$, a set $E \subseteq AR$ and a strongly connected component $S \in SCCS_{AF}$. We will present the definition of some useful sets, the formal definition of these sets is in [1]. The set $D_{AF}(S, E)$ consists of the nodes of S attacked by E from outside S, the set $U_{AF}(S, E)$ consists of the nodes of S that are not attacked by E from outside S and are defended by E (i.e., their defeaters from outside S are all attacked by E), and $P_{AF}(S, E)$ consists of the nodes of S that are not attacked by E from outside S and are not defended by E (i.e., at least one of their defeaters from outside S is not attacked by E). Finally, $UP_{AF}(S, E) = (S \setminus D_{AF}(S, E)) = (U_{AF}(S, E) \cup P_{AF}(S, E))$.

Here, we define $GF(AF, C)$ for an argumentation framework $AF = \langle AR, attacks \rangle$ and a set $C \subseteq A$, representing the defended nodes of AF: two cases have to be considered in this respect.

If AF consists of exactly one strongly connected component, it does not admit a decomposition in which can be applied the directionality principle, therefore it has to be assumed that $GF(AF, C)$ coincides in this case with a *base function*, denoted as

$BF_S(AF, C)$, that must be assigned in order to characterize a particular argumentation semantics S.

On the other hand, if AF can be decomposed into several strongly connected components, then, $GF(AF, C)$ is obtained by applying recursively GF to each strongly connected component of AF, deprived of the nodes in $D_{AF}(S, E)$. Formally, this means that for any $S \in SCCS_{AF}$, $(E \cap S) \in GF(AF \downarrow_{UP_{AF}(S,E)}, C')$, where C' represents the set of defended nodes of the restricted argumentation framework $AF \downarrow_{UP_{AF}(S,E)}$. The set C' can be determined by taking into account both the attacks coming from outside AF and those coming from other strongly connected components of AF.

Definition 8 *[1] A given argumentation semantics S is* SCC-recursive *if and only if for any argumentation framework $AF = \langle AR, \text{attacks}\rangle$, $E_S(AF) = GF(AF, AR)$, where for any $AF = \langle AR, \text{attacks}\rangle$ and for any set $C \subseteq AR$, the function $GF(AF, C) \subseteq 2^{AR}$ is defined as follows: for any $E \subseteq AR$, $E \in GF(AF, C)$ if and only if*

- *in case $|SCCS_{AF}| = 1$, $E \in BF_S(AF, C)$,*
- *otherwise,$\forall S \in SCCS_{AF}(E \cap S) \in GF(AF \downarrow_{UP_{AF(S,E)}}, U_{AF}(S, E) \cap C)$.*

where $BF_S(AF, C)$ is a function, called base function, that, given an argumentation framework $AF = \langle AR, \text{attacks}\rangle$ such that $|SCCS_{AF}| = 1$ and a set $C \subseteq AR$, gives a subset of 2^{AR}.

Definition 8 does not define any particular semantics, but defines a general structure, where if one changes the base function, $BF_S(AF, C)$, one can induce several semantics. In particular, when $BF_S(AF, C)$ is the function that returns all sets that are free of maximal conflicts, $max\_conflict\_freeSets(AF)$, then $CF2$ is obtained.

1.3. Extension-based Argumentation Semantics via Labeling

The labeling process is based in a labeling mapping. In this paper, we are going to consider the mapping presented by Caminada in [7]. A labeling mapping is a set of labels (one per argument in the framework) where each label can represent three different values: *IN*, *OUT* and *UNDEC*. This mapping is defined as follows: Given an argumentation framework $AF = \langle AR, attacks\rangle$, for any argument $a \in AR$ one can define functions $IN(a)$, $OUT(a)$ and $UNDEC(a)$ that return true if the argument a is labeled IN, OUT and $UNDEC$ respectively, and false otherwise. IN, OUT and $UNDEC$ also represent the sets of arguments labeled IN, OUT and $UNDEC$ respectively.

Some authors have shown that by considering a mapping process one can characterize the extension-based argumentation semantics defined by Dung in [4] and also some other new argumentation semantics that have been defined by the argumentation community.

2. WizArg software

The WizArg software aims to provide generic Java processes with argumentation-based reasoning capabilities. The WizArg project allows generic software processes to use argumentation meta-interpreters, effectively enabling them to solve argumentation frameworks using a wide range of argumentation semantics. By solving argumentation frame-

Figure 2. WizArg Argumentation library example: framework solving

works, generic software components can show intelligent behavior, such as: a) negotiation [13] including skeptical and credulous reasoning b) knowledge refination [8], including justification of information and processes to deal with dynamic on incomplete information. The WizArg project consists in two main components: 1) The WizArg Argumentation library, that can be used by any generic software to solve argumentation frameworks and answer questions about them, such as: is an argument credously/skeptically accepted in a given framework for a given semantics? 2) The WizArg front-end, a generic software component that allows users to create, save, and load their own argumentation frameworks. Argumentation frameworks can be processed by the Argumentation library, enabling users to visualize the framework solutions and interactively ask questions about them.

2.1. WizArg Argumentation library

The WizArg Argumentation library, makes use of the DLV wrapper project [12] to invoke the logic programs (which characterize the extension-based argumentation semnatics) that act as argumentation meta-interpreters for the different semantics implemented. The WizArg library hides the invocation details to the user, enabling an easy and fast access to the different argumentation meta-interpreters. The WizArg library provides two main capabilities: Framework solving and question answering.

The framework solving capability is performed by the *SolveFramework* operation of the *FrameworkSolver* class. This operation takes as inputs: a) a file with an argumentation framework, b) filter (typically *'IN,OUT,UNDEC'*) c) the name of the semantics to be used. The operation will return a set of answers, that is, possible solutions for the tuple framework - semantics specified, filtered via the filter parameter. Each of this answers contains a set of labeled arguments representing the solution. The labels used are based on the ones defined in Section 1.3, that is, IN for accepted arguments, OUT for defeated arguments and UNDEC for the ones that are undecidable. Figure 2 shows an example of the invocation of the SolveFramework function, including: 1) graphical representation of the framework used in the example, 2) representation of this framework on the file used as input. 3) Java code used for invoking the function, 4) graphical representation of the result returned by the SolveFramework function, including three possible solutions with three labeled arguments each.

Please, notice that, given an argumentation framework $AF = \langle AR, attacks \rangle$ the input file includes a representation of the arguments of the form $ar(X)$ such that $X \in$

AR and a representation of the attack relations of the form $at(X, Y)$ such that $(X, Y) \in attacks$.

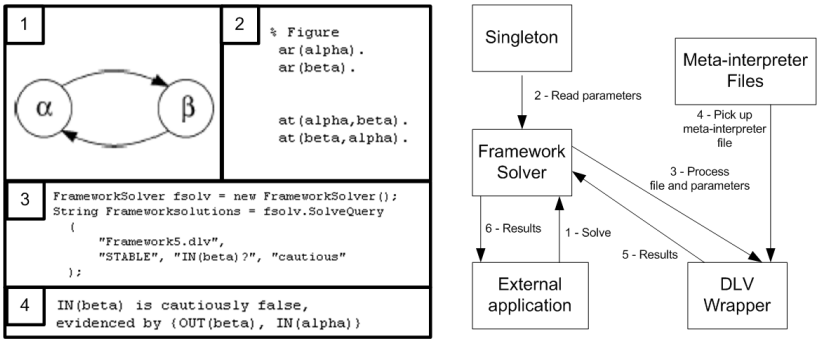

Figure 3. WizArg Argumentation library question answering example and WizArg architecture

The question answering capability is performed by the *SolveQuery* operation of the *FrameworkSolver* class. This operation takes as inputs: a) a file with an argumentation framework, b) the name of the semantics to be used, c) the question to be answered, that is, a label (IN,OUT or UNDEC) applied to an argument in the framework (e.g. *IN(x)?*) d) the type of reasoning to be performed, either *brave* or *cautious*[3]. The operation will return a text String with the answer to the question. Figure 3 shows an example of the invocation of the SolveQuery function, including: 1) graphical representation of the framework used in the example, 2) representation of this framework on the file used as input. 3) Java code used for invoking the function, 4) result returned by the function, including evidence to support it.

Figure 3 also shows the information flow of an invocation to the WizArg Argumentation library. First, an *External application* invokes the argumentation library (1) for solving an argumentation framework or answering questions about it for a given semantics. Then, the *Framework Solver* class gathers parameters about the semantics via the Singleton class(2). This parameters include the name of the dlv meta-interpreter file to be used. Later, the *Framework Solver* invokes the *DLV Wrapper* (3) with the parameters gathered both from the invocation and the *Singleton* class. It is time for the *DLV Wrapper* to solve the framework, picking up the meta-interpreter to be used from the set of *Meta-interpreter files* (4). The associations semantic - metainterpreter are stored on the *Singleton* class. Then, the results are returned to the *Framework Solver*(5) that finally passes them to the *External Application* (6).

WizArg is an open-source project, and thus, advanced users can modify its code adapting it to their needs. One of the most immediate adaptations one can think of is adding new meta-interpreters to support new argumentation semantics or modify the existing ones. The WizArg library presents an architecture that makes this modifications easy and immediate. Modifying an existing meta-interpreter is as easy as editing the associated dlv file. This operation can be done even on the fly, while a program using the WizArg library (such as the WizArg front-end) is running. Adding a new meta-interpreter

[3] A fact is *bravely* true if it is true in at least one of the possible models of the world. A fact is *cautiously* true if it is true in all possible models of the world

requires following these steps: a) create a new dlv file for the meta-interpreter b) Add the parameters of the new semantics to the Singleton file, modifying functions *translateSemantics* (to enable the WizArg front-end to use this semantics), *initFileSemanticsMap* (associating the semantics to the meta-interpreter file) and *initDeterministicSemanticsMap* (typically, setting it as non deterministic semantics). Future versions of the project will include a meta-interpreter configuration file to make this process even easier.

2.2. WizArg front-end

The WizArg front-end makes use of the WizArg Argumentation library and the Jung[4] libraries. The WizArg library provides means to effectively solve the argumentation frameworks specified by the user, and answer questions about them. The Jung libraries allow the user to graphically design their own argumentation frameworks (or modify the ones provided along with the WizArg software) and show the results of the solving process.

The WizArg front-end takes an approach based on labeling (see *Section* 1.3) for representing the argumentation frameworks graphically. Given an argumentation framework $AF = \langle AR, attacks \rangle$, WizArg is drawing a directed graph of the form $Graph = \langle Nodes, Edges \rangle$ where the nodes are the arguments (i.e. $Nodes = AR$) and there is a directed edge between two arguments when there is an attack relation between them (i.e. $\forall X, Y \in AR \wedge \in Edges : \langle X, Y \rangle \in attacks$ *iff* $\langle X, Y \rangle \in Edges$). Notice that for any element in the $Edges$ set the first node represents the origin node (corresponds to attacker argument on Argumentation Framework) and the second one, destination node (corresponds to attacked argument on Argumentation Framework).

When solving argumentation frameworks, the WizArg front-end will paint the nodes following a labeling approach, applying the following colors to each of the nodes: a) red if the argument represented by the node is defeated (corresponds to OUT label) b) green if the the argument represented by the node is accepted (corresponds to IN label) c) yellow if the argument represented by the node is undecidable (corresponds to UNDEC label).

When designing new argumentation frameworks or modifying existing ones, user can add and delete nodes or attack relations, and move existing nodes (making the graph that represents the framework easier to understand visually). When visualizing results, user is not allowed to add or delete elements, but can freely move existing ones. What's more, user can make use of a wide range of graph drawing algorithms to automatically order the elements on the graph.

Figure 4 shows the different windows available on the WizArg front-end: 1) Main windows where user can choose to edit argumentation frameworks (a) or solve them (b) by specifying the framework (c) and the semantics to be used (d) 2) Framework editing window where users can define arguments and attack relations on the editing zone (e) (clicking on the editing zone will create an argument, clicking on an argument will create an arrow that can be dropped on an argument effectively creating an attack relation), import existing frameworks to modify them (f) and save the actual framework to a format the WizArg library can process (g) 3) Framework solving window where users can see the results of the framework solving on the solving zone (h), re-draw the framework to visualize it better (i), check the different solutions available (j) and ask several questions regarding the framework (k).

[4] http://jung.sourceforge.net/index.html

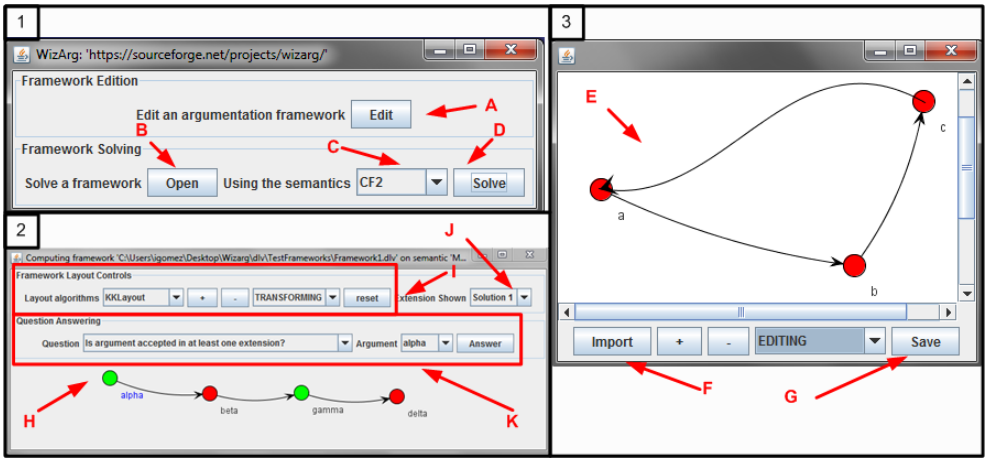

Figure 4. WizArg front-end example

3. Conclusions and future work

The Dung's argumentation style, introduced in [4] and extended by several authors [1,10], is a versatile and a powerful tool for defeasible reasoning. However, one can accept that there is a lack of open source planforms/tools for implementing extension-based argumentation systems. This lack of friendly libraries supporting argumentation inference bases on extensions-based argumentation semantics increase the gap between theoretical argumentation results and extension-based argumentation systems. Possibly one of the main reasons of this lack of general purpose libraries for supporting argumentation inference is the fact that the computational complexity of argumentation inference is hard, see Figure 1.

In this paper, we introduce a general purpose library for supporting and exploring extension-based argumentation semantics, this library is called WizArg. The WizArg project allows generic software processes to use argumentation meta-interpreters, effectively enabling them to solve argumentation frameworks using a wide range of argumentation semantics. The current version of WizArg gives support to the grounded, stable, greferred and CF2 argumentation semantics in order to give answer to questions of kind presented in Figure 1. The WizArg project consists in two main components: 1) The WizArg Argumentation library, that can be used by any generic software to solve argumentation frameworks and answer questions about them, 2) The WizArg front-end, a generic software component that allows users to create, save, and load their own argumentation frameworks. Argumentation frameworks can be processed by the Argumentation library, enabling users to visualize the framework solutions and ask questions about them.

A line of future work is on integrating the WizArg library on Multi-Agent platforms based on Java Code (such as AgentScape[5] and PAWS[6]). This would allow agents to perform argumentation based reasoning in order to solve different issues. Future lines of work include tackling two main problems via argumentation based reasoning:

[5]http://www.iids.org/research/aos
[6]http://sourceforge.net/projects/paws-ai/

1) Distributed plan agreement. When a global plan, divided into different sub-plans has to be enacted collaboratively between a set of intelligent agents, and each of this agents can choose among a set of different alternatives to enact its sub-plan. Agents can play arguments in favor or against the enactment of a certain sub-plan alternative and argumentation-based reasoning can help to reach a justified global agreement about which alternatives have to be chosen in order to collaboratively enact the global plan. 2) Meta-reasoning. When a given agent has several reasoning modules available (e.g. CBR, inference-based reasoning, etc.) means to decide which one of them will be used on a given situation are required. Exploring how argumentation based reasoning can help in providing a feasible and efficient way of taking this decision even in the presence of dynamic and uncertain information (a flexible alternative to the static rule-based process used on most meta-reasoning methods) is an open line of future work.

It must be remarked the authors want the WizArg project to be an active project. The most important line of future work is to keep the WizArg project alive, extending and improving int as the community using it requires so.

Acknowledgements

We are grateful to anonymous referees for their useful comments. This research has been partially supported by the EC founded project ALIVE (FP7-IST-215890). The views expressed in this paper are not necessarily those of the ALIVE consortium.

References

[1] P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168:162–210, October 2005.

[2] T. J. M. Bench-Capon and P. E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15):619–641, 2007.

[3] S. DLV. Vienna University of Technology. http://www.dbai.tuwien.ac.at/proj/dlv/, 1996.

[4] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[5] P. E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artificial Intelligence*, 171(10-15):701–729, 2007.

[6] A. J. García and G. R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2):95–138, 2004.

[7] S. Modgil and M. Caminada. *Argumentation in Artificial Intelligence*, chapter Proof Theories and Algorithms for Abstract Argumentation Frameworks, pages 105–129. Springer, 2009.

[8] D. P. N. Karacapilidis and T. Gordon. An argumentation based framework for defeasible and qualitative reasoning. In *Advances in Artificial Intelligence*, volume 1159/1996 of *Lecture Notes in Computer Science*, pages 1–10. Springer-Verlag Berlin Heidelberg, 2006.

[9] J. C. Nieves, M. Osorio, and U. Cortés. Preferred Extensions as Stable Models. *Theory and Practice of Logic Programming*, 8(4):527–543, July 2008.

[10] J. C. Nieves, M. Osorio, and C. Zepeda. A Schema for Generating Relevant Logic Programming Semantics and its Applications in Argumentation Theory. *Fundamenta Informaticae*, accepted, 2010.

[11] M. Osorio, J. C. Nieves, and I. Gómez-Sebastiá. CF2-extensions as Answer-set Models. In *the Third International Conference on Computational Models of Argument COMMA'2010*.

[12] F. Ricca. A java wrapper for dlv. *Answer Set Programming*, 2003.

[13] M. Schroeder. An efficient argumentation framework for negotiating autonomous agents. In *Multi-Agent System Engineering*, volume 1647/1999 of *Lecture Notes in Computer Science*, pages 140–149. Springer-Verlag Berlin Heidelberg, 2007.