An Argumentation Framework for BDI Agents

Tudor Berariu

Abstract. This article presents a practical approach to building argumentative BDI agents. As in the last years the domain of argumentation reached maturity and offers now a very rich and well structured abstract theory, the challenge now is to put this work into practice and prove its usefulness in real applications. There is a high interest from the multi-agent systems community in applying argumentation for agents' defeasible reasoning.

The main goal of the work presented in this paper was to provide the means to enable argumentative capabilities in BDI agents. For this reason, Jason, a platform for the development of multi-agent systems using the BDI model of agency, was extended with a module for argumentation. The proposed argumentation module is decoupled from the BDI reasoning cycle as it operates only on the belief base of the agents and does not interfere in the execution of plans, creation of goals, or agent's commitments. Although no protocol for argumentation-based dialogues is proposed here, agents can engage in any such dialogues as the argumentation module makes suggestions of attacks to put forward in conversation or gives structured justifications for different beliefs. An instantiation of Dung's abstract framework is used with state of the art structure of arguments and ways of attack and defeat between arguments.

1 Introduction

One fundamental aspect in artificial intelligence and multi-agent systems is agent reasoning about the external world, itself or the actions to take at a certain point of time.

One common problem in agent logic is ensuring the consistency of beliefs after information conflicting with previous view of the world is perceived. After such

Tudor Berariu
University Politehnica of Bucharest, Bucharest, Romania
e-mail: tudor.berariu@gmail.com

F. Zavoral et al. (eds.), *Intelligent Distributed Computing VII,*
Studies in Computational Intelligence 511,
DOI: 10.1007/978-3-319-01571-2_40,

an information update, the consistency of the belief set must be preserved and this process is called *non-monotonic reasoning*. Two distinct approaches to its formalization are known. First, there are different extensions to the classical logic like McDermott and Doyle's modal operator **M** [12], Reiter's logic for default reasoning [14] or McCarthy's circumscription theory [11]. The second class of formalisms for mechanizing non-monotonic reasoning are the various families of truth maintenance systems (TMS), first proposed by Doyle in [6].

Close to Doyle's vision, argumentation provides an alternative way to deal with non-monotonic reasoning: arguments can support existing beliefs or can act as counterarguments against them. In this way, solving conflicts between arguments does the job of belief revision.

The paper presents an argumentation-based system able to maintain consistency of the belief base of BDI agents, while hiding the functioning of the argumentation mechanisms from the BDI reasoning cycle. The system allows the agents to query the argumentation module for suggestions of attacks or argumentation-based justifications for accepted or rejected beliefs

The approach starts from Dung's abstract argumentation framework [7] but it is centered towards the integration of argumentation with the practical aspects of BDI agent behaviour by developing an argumentation module to be integrated in the Jason platform[1].

The document is structured as follows: section 2 introduces some fundamental concepts from argumentation theory, section 3 describes my practical solution to the problem stated above, while section 4 brings an example of non-monotonic reasoning that uses argumentation in the proposed system.

2 Abstract Argument Systems

The work of Dung [7] is considered the first major step towards *argumentation systems* as it provides the means to use argumentation theory for non-monotonic reasoning. Dung offers an abstraction of the attack relation between arguments, on top of which refutation, a central concept in argumentation, is built. A *refutation* of an argument is an opposed argument that attacks the original argument and defeats it. There are several ways to attack an argument: by asking an appropriate critical question that raises doubt about the acceptability of the argument, by questioning one of its premises or by putting forward counter-arguments that oppose the original argument, meaning that the conclusion of the opposing argument is the opposite (negation) of the conclusion of the original argument. There are also more complex ways to attack arguments: doubting about the relevance of the premises to the conclusions or even about the relevance of the argument in relation to the issue of the dialogue, arguing that a set of arguments commit a logical fallacy.

An example of Dung-style representation of arguments is that shown in Figure 1. In that example, argument $A1$ attacks $A2$, $A3$ and $A4$; $A3$ attacks $A5$; $A5$ and $A6$ attack each other.

[1] `http://jason.sourceforge.net/Jason/Jason.html`

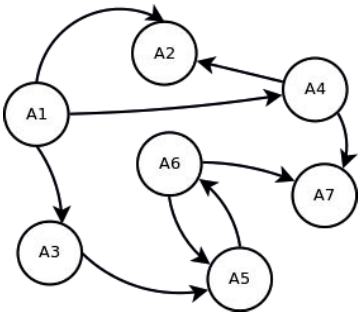


Fig. 1 Representation of argument attacks

An *abstract argument system* is a tuple $\langle \mathscr{A}, \mathscr{R} \rangle$ where $\mathscr{A}$ is a set of *arguments* and $\mathscr{R}$ is a binary relation over $\mathscr{A}$ called *attack relation*.

It is clear that, while arguments attack each other, they cannot stand together and their status is subject to *evaluation*. That means that the *justification state* of each argument must be determined. An argument is regarded as *justified* if it survives the attacks it receives and it is *rejected* otherwise.

The *argument evaluation* needs a formal method that describes the steps of the process or the states of arguments based on some criteria. These formal methods are called *argumentation semantics* and there are two categories: *extension-based* and *labeling-based*.

Extension-based semantics specifies how to obtain sets of arguments $\mathscr{E}$ where each extension E of an argumentation framework $\langle \mathscr{A}, \mathscr{R} \rangle$ is a subset of $\mathscr{A}$ containing a set of arguments that can stand together. In extension-based semantics the *justification state* of an argument is defined in terms of membership of the respective argument to the extensions.

In extension-based semantics two alternative types of justification, namely *skeptical* and *credulous* can be considered. In a formal way, for an argumentation framework AF and a semantics $\mathscr{S}$, an argument a is:

- *skeptically justified* if and only if $\forall E \in \mathscr{E}_{\mathscr{S}}(AF) : a \in E$
- *credulously justified* if and only if $\exists E \in \mathscr{E}_{\mathscr{S}}(AF) : a \in R$.

Using this classification, *justification states* of arguments can be defined. An argument a is:

- *justified* if and only if $\forall E \in \mathscr{E}_{\mathscr{S}}(AF) : a \in E$ (this corresponds to skeptical justification);
- *defensible* if and only if $\exists E_1, E_2 \in \mathscr{E}_{\mathscr{S}}(AF) : a \in E_1, a \notin E_2$ (this corresponds to credulous justification)
- *overruled* if and only if $\forall E \in \mathscr{E}_{\mathscr{S}}(AF) : a \notin E$ (arguments that cannot be justified are rejected).

Building on definitions for acceptable arguments (that are defended against all attacks) and admissible sets (that contain only acceptable arguments), Dung proposes four *traditional* semantics:

complete E is a complete extension if and only if E is admissible and every argument of $\mathscr{A}$ which is acceptable wrt. E belongs to E

grounded The grounded semantics are easier to explain by the process of building them incrementally from the unattacked arguments. The arguments attacked by them can be suppressed. The process is repeated until no new arguments arise after a deletion step. The set of all initial arguments identified so far is the grounded extension.

stable A stable extension attacks all arguments not included in it.

preferred An extension E is a *preferred extension* of AF if E is as large as possible and able to defend itself from attacks.

There are other extension-based semantics proposed in the literature: *stage semantics* [15] (the stage extension is the maximum conflict-free set); *semi-stable semantics* [4] (the semi-stable extension is the maximal complete extension); *ideal semantics* [8]; *CF2 semantics* [1]. Another semantics proposed, the *prudent semantics* [5] is based on a more extensive notion of attack in the context of traditional semantics: an argument *a indirectly attacks* an argument *b* if there is an odd-length attack path from *a* to *b*. Recent work on argumentation semantics explore methods of local computation of extensions [10].

This represents the theoretical basis for our work.

3 Building Argumentative Agents in Jason

There are some attempts in the literature to combine the BDI model of agency with argumentation based reasoning. Most of these works concern negotiation or other dialogue games where agents have to respect a certain protocol. The challenge tackled in this document is a bit different. The goal here is to build an argumentation module attached to the BDI engine that enables general argumentation capabilities to the agents, not just for a specific dialogue game. By this claim I mean that an agent capable of a non-monotonic argumentation based reasoning can participate into argumentation dialogues if the rules to follow a specific protocol are programmed in the agent and, possibly, an argumentation strategy is defined. For persuasive agents, such a strategy should work using a proper argumentation semantics that identifies the set of arguments to defeat, as the one proposed in [9]. An aspect one should be careful when designing an argumentation based reasoning engine for a BDI agent is that information comes from multiple sources which can influence the status of each piece of information from an argumentation point of view and, as a consequence, the entire reasoning process.

The framework used in this project is based on the latest instantiations of Dung's abstract formalism, especially on Prakken's work [13]. As in [13], I used an abstract argumentation framework with structured arguments and three types of attacks

between them: rebutting, undercutting and undermining. Also, the framework takes advantage of the distinction between contradiction and contrariness (as in [2]).

To construct multi-agent systems in the BDI paradigm, we chose Jason a platform that uses a high level language (an extension of AgentSpeak) [3] for programming the agents.

The approach taken in this work was to separate the BDI *strict* reasoning cycle (the Jason reasoning cycle) from the argumentation *defeasible* reasoning. The result is that there are two reasoning modules that operate on the agent's knowledge, but not on the same piece of information. This might seem a disadvantage at first, but there are rational reasons to do that.

One advantage of decoupling the BDI reasoning module from the argumentation module is the fact that agents might receive (sense) a lot of data that is irrelevant from an argumentative point of view. There are beliefs which generate goals and plans that do not need an argumentation treatment. For example, an agent has a motion sensor and its only use is decide whether to turn on the light or not. Representing all this data as arguments in an argumentation framework brings an unneeded overhead.

3.1 Coupling the Modules

When a new belief is added, deleted or a new set of percepts are received from the environment, before the BDI logic starts to treat the new events, these are intercepted by the argumentation module. Here a sequential update on many layers is done and a set of visible modifications (additions and removals) to the belief base is sent to the BDI reasoning engine in Jason.

First, from the set of all beliefs and percepts received (for either addition or removal), only those that are relevant for the argumentation system are kept. The others are passed untouched to the BDI engine. The filtering is done on the basis of the language and on the formulas that appear in rules. If there is no rule that has as an antecedent or as its consequent a certain formula, then the latter is not relevant for argumentation. This verification adds a small computational overhead for the cases when beliefs or percepts that are irrelevant to argumentation pass through the argumentation module.

Next, these beliefs and percepts are transformed into premises in the knowledge base of the argumentation theory. Here, the type of premise is decided as described later. There are several rules that apply in order (user expressed preferences, custom functions or default rules from the argumentation theory: e.g. the Carneades model).

On the basis of the modified premises (new premise, premises whose types were changed, deleted premises) the set of matched rules is also updated. New arguments are formed if new rules match or old arguments are deleted if they correspond to premises that were removed.

At the next level, the list of attacks between arguments is updated. First, attacks from and to deleted arguments are removed. Second, new attacks for new arguments based on the contrariness function are computed.

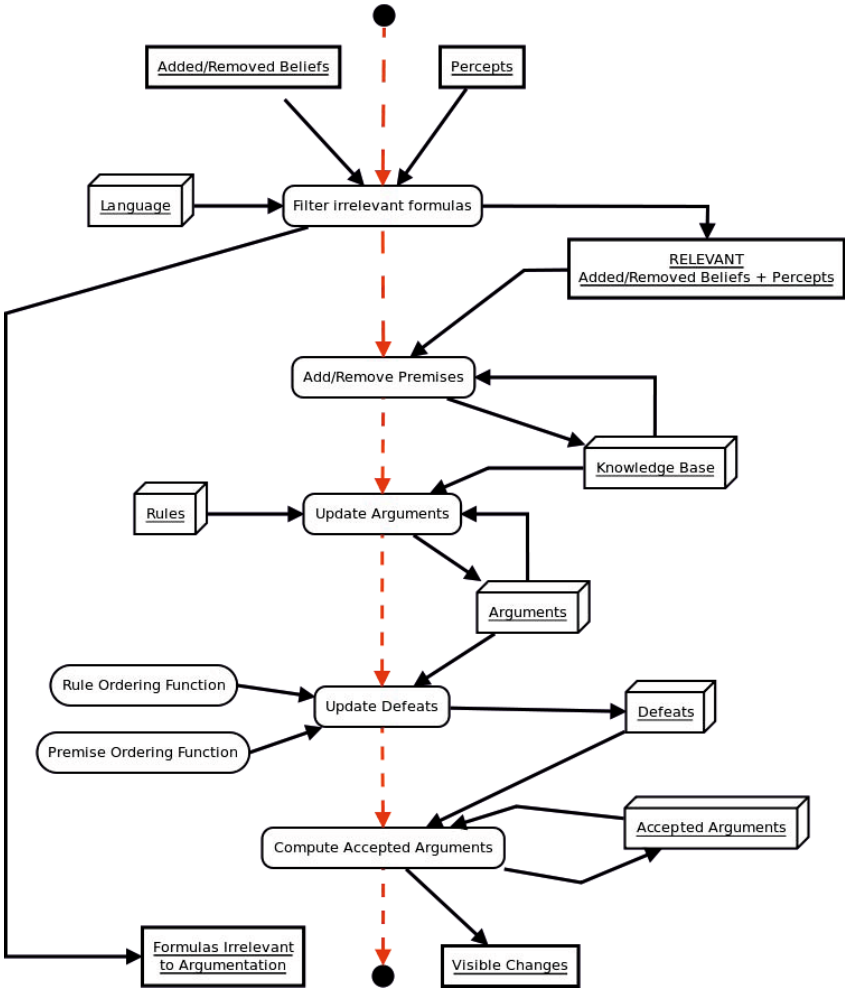


Fig. 2 The course of action in the argumentation framework when a new belief is added/removed or perceived

With the new list of attacks, the successful attacks which result in defeat are filtered. This is done using the theory described in the previous section.

The updates propagate further to the semantic extensions of arguments. The last set of accepted arguments is saved. With the new graph of defeats between arguments a new set of acceptable arguments is computed. The preferred semantics are used as they have the highest level of consistency. The differences between the old set and the new set of accepted arguments are actually the changes that are transmitted back to the Jason agent to be processed by the agent's belief revision function.

3.2 Controlling the Argumentation Module through Beliefs

As all added or removed beliefs pass through the argumentation framework, the same strategy was used to control the argumentation module with beliefs that have reserved predicate names.

In order to control the way in which conclusions are accepted, I introduced the `argument_acceptability(arg)` belief, where `arg` takes on of the following values: {`CREDULOUSLY_ACCEPTABLE` for *credulously S-acceptable* conclusions, `SKEPTICALLY_ACCEPTABLE`, or `RANDOM`}. In a similar manner, using the `plausible_argument_orderings` one can configure one of the two plausible argument orderings defined by Amgoud: `LAST_LINK` or `WEAKEST_LINK`.

There are two special beliefs for defining strict and defeasible rules. `defeasible_rule(RuleName,RuleText)` adds a defeasible rule to the argumentation system, while `strict_rule(RuleName,RuleText)` adds a strict rule. Rules are given in the following format:

$$literal_1 \& \ldots \& literal_n \Rightarrow \varphi$$

where φ is the conclusion and $literal_i$ with $i \in \{1,\ldots,n\}$ are antecedents.

Next, there are two predicates used to define the contradiction and the contrariness relations: `contradictory(Literal1, Literal2)` and `contrary(Literal1, Literal2)`. As undercutting attacks against the appliance of a rule are needed, beliefs in the form `evidence_against(Literal, RuleName)` add such information in the argumentation system.

In order to differentiate between different types of premises based on their source, the `premise_from(AgentName,PremiseType)` beliefs can be used.

As the argumentation module should help the agent in persuasion, inquiry, negotiation or deliberation dialogues, another belief that queries the argumentation module has been added. `why(Proposition)` interrogates the argumentation module to find out why the respective argument is accepted or rejected. The response comes in a belief `because(X,Y)` where `X` is one of $\{in, out\}$ and possible answers are:

- `because(out,unknown)` : which means that the proposition in the query is not in the knowledge base. Agent is not aware of the query formula or its negation.
- `because(in,premise(premise_type))` : which means that the specific proposition was added as a premise in the knowledge base and is not the result of any form of reasoning (strict or defeasible). The current status of that premise is `premise_type` (axiom, assumption,...).
- `because(out,¬Proposition)` which means that the formula in the query is not itself in the knowledge base, but its negation is currently an accepted argument.
- `because(in,¬Proposition)` which means that the formula in the query is not itself in the knowledge base, but its negation is currently an overruled argument.

- `because(in,Rule)` : which tells the agent that the argument corresponding to the proposition in the query is currently accepted and it is the result of a applying the (defeasible or strict) rule `Rule`.
- `because(out,ListOfDefeats)` : which returns a list with all the conclusions of the arguments that defeated the current one.

The argumentation engine maintains a graph-like representation for the arguments and their justifications. In order to resolve `why(Proposition)` queries, the arguments space is explored starting from the belief matching `Proposition`.

4 Case Study: Business Trip

In what follows a scenario for argumentation implemented in Jason is described. This example uses a single agent for whom the argumentation module works just as a nonmonotonic reasoning agent.

In the next figures, a green box represents an accepted argument, a red box a refuted one, while blue is for strict rules and yellow for defeasible rules.

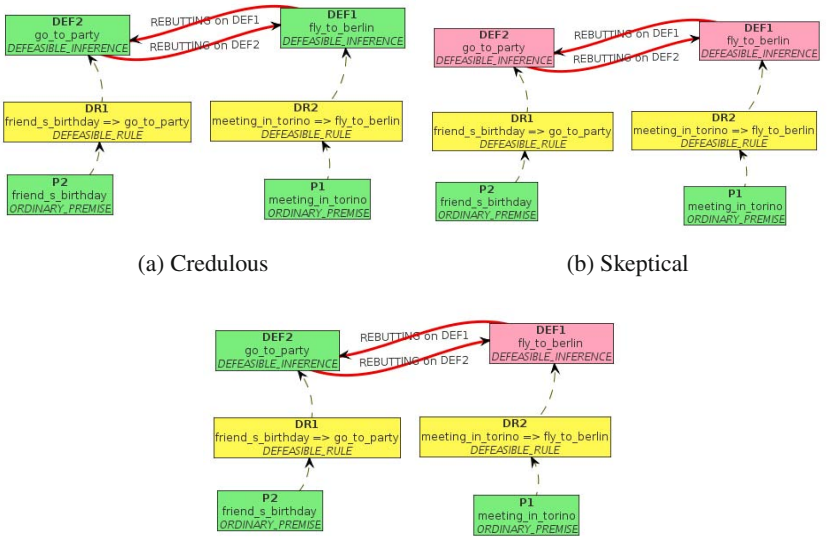
Consider an agent that has only one belief that represents the information that is the birthday of one of his friends and one defeasible rule which says that if is the birthday of a friend, then he probably goes to a party. Consider now that we add another belief to the agent. Suppose he has a meeting in Berlin, so he will probably fly to Berlin. He cannot be both in Berlin and Bucharest at the same time, so the two propositions are marked as contradictory.

```
friend_s_birthday.
defeasible_rule("DR1","friend_s_birthday => go_to_party").
meeting_in_berlin.
defeasible_rule("DR2","meeting_in_berlin => fly_to_berlin").
contradictory("go_to_party","fly_to_berlin").
```

Now, arguments for `go_to_party` and `fly_to_berlin` attack each other. As there are no preferences between defeasible rules or premises that can be applied here, both attacks are successful.

There are two preferred extensions, one that contains `fly_to_berlin` and one that contains `go_to_party`. Hence, if the agent is credulous, he will accept both and if he is a skeptical agent, he will accept none of the two (see Figure 3). Both provide good information about the uncertainty of arguments, but neither is useful from a practical point of view. An agent cannot use all the arguments that he credulously accepted, as there can be pairs of conflicting arguments and it's not useful to reason just on the arguments that can be skeptically accepted as that could lead to no action taken (not going to the party and not flying to Berlin).

Now, let's consider, that in general, if you have to go to a business meeting you go in most of the cases. But you are not going to all of your friends' birthday parties. So, we add a rule that says that the appliance of rule `DR2` is more probable than the appliance of rule `DR1`.



(a) Credulous

(b) Skeptical

(c) Random preferred

Fig. 3 Accepted arguments depending on the acceptability principle

```
argument_ordering("LAST_LINK").
prefer_rule("DR1","DR2").
```

Now, as just the attack from `DEF1` to `DEF2` succeeded, there is only one preferred extension (so all arguments are both skeptically and credulously accepted). The new state of the arguments is represented in Figure 4.

Now, suppose that the agent watched the news were the possibility of another eruption of the volcano in Island was announced. This would mean that the airports will be closed. This last argument provides a situation in which the rule `meeting_in_berlin` $\Rightarrow$ `fly_to_berlin` cannot be applied. That results in an undercutting attack as in Figure 5.

The Jason code for the information added is:

```
volcano_at_news.
evidence_against("airport_closed","DR2").
strict_rule("SR1","volcano -> airport_closed").
defeasible_rule("DR3","volcano_at_news => volcano").
```

As the argument corresponding to `airport_closed` is not attacked by any other argument, it will defeat the argument with the conclusion `fly_to_berlin` in all preferred extensions.

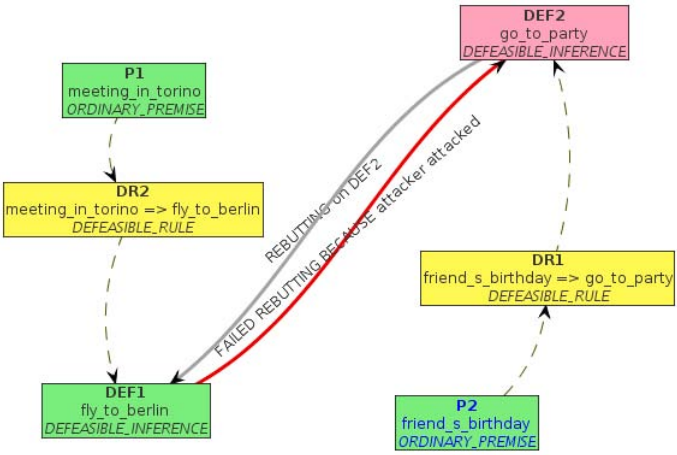


Fig. 4 Argument ordering using rule preferences

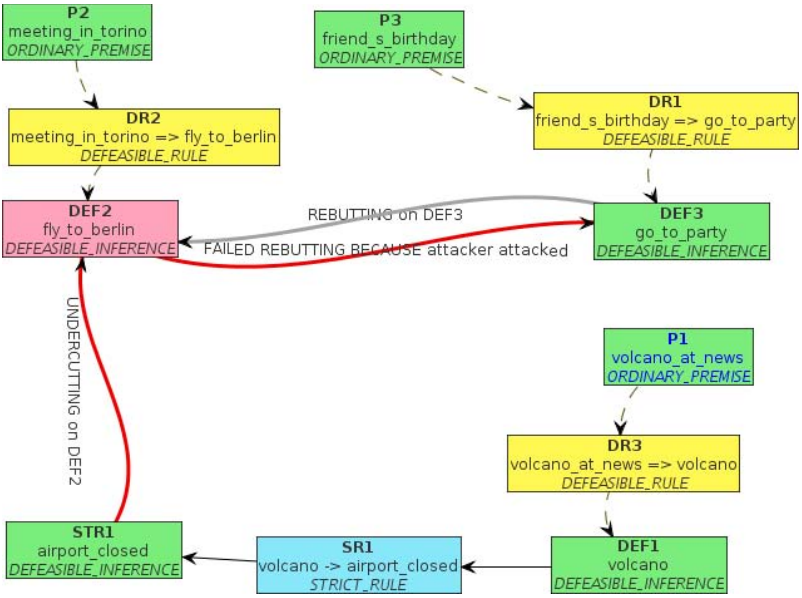


Fig. 5 Example of undercutting attack

5 Conclusions and Future Work

This paper presents a framework for supporting abstract argumentation reasoning in BDI agents. State-of-the art concepts from argumentation theory were put in

practice using a popular platform for developing BDI agents, Jason. In our work, we implemented a module to be used when programmers want to add argumentation reasoning to their agents built in Jason. The use of this module does not affect the rest of the functioning of Jason as the module intercepts any addition or removal of beliefs in forms of percepts , messages from other agents or mental notes of the agent itself and outputs to the Jason reasoning engine the effects of applying argumentation to that specific change. What the agent will be aware of (in the Jason context) is a set of consistent beliefs, as the other pieces of information corresponding to defeated arguments are hidden.

Future work will investigate the impact argumentation has on different types of dialogues between agents, especially in negotiations. Another line of development we foresee emerging from this work is the study of the utility of different semantics in practical scenarios of multi-agent interaction.

Acknowledgment. This work has been funded by project ERRIC (Empowering Romanian Research on Intelligent Information Technologies), number 264207/FP7-REGPOT-2010-1.

References

[1] Baroni, P., Giacomin, M., Guida, G.: SCC-recursiveness: a general schema for argumentation semantics. Artificial Intelligence 168(1-2), 162–210 (2005)
[2] Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial intelligence 93(1), 63–101 (1997)
[3] Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley and Sons, Ltd. (2007)
[4] Caminada, M.: Semi-stable semantics. In: Computational Models of Argument: Proceedings of COMMA, pp. 121–130 (2006)
[5] Coste-Marquis, S., Devred, C., Marquis, P.: Prudent semantics for argumentation frameworks. In: 17th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2005, pp. 5–572. IEEE (2005)
[6] Doyle, J.: A truth maintenance system* 1. Artificial Intelligence 12(3), 231–272 (1979)
[7] Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games* 1. Artificial Intelligence 77(2), 321–357 (1995)
[8] Dung, P., Mancarella, P., Toni, F.: A dialectic procedure for sceptical, assumption-based argumentation. In: Proceeding of the 2006 conference on Computational Models of Argument: Proceedings of COMMA 2006, pp. 145–156. IOS Press (2006)
[9] Gratie, C., Florea, A.M.: Argumentation semantics for agents. In: Cossentino, M., Kaisers, M., Tuyls, K., Weiss, G. (eds.) EUMAS 2011. LNCS, vol. 7541, pp. 129–144. Springer, Heidelberg (2012)
[10] Gratie, C., Florea, A.M., Meyer, J.J.C.: General directionality and the local behavior of argumentation semantics. In: Ossowski, S., Toni, F., Vouros, G.A. (eds.) Proceedings of the First International Conference on Agreement Technologies, AT 2012, Dubrovnik, Croatia, October 15-16. CEUR Workshop Proceedings, vol. 918, pp. 113–127. CEUR-WS.org (2012)
[11] McCarthy, J.: Circumscription–a form of non-monotonic reasoning. Artificial Intelligence 13(1-2), 27–39 (1980)

[12] McDermott, D., Doyle, J.: Nonmonotonic logic 1. Artificial Intelligence 13, 41–72 (1980)
[13] Prakken, H.: An abstract framework for argumentation with structured arguments. Argument & Computation 1(2), 93–124 (2010)
[14] Reiter, R.: A logic for default reasoning. Artificial Intelligence 13(1-2), 81–132 (1980)
[15] Verheij, B.: Two approaches to dialectical argumentation: admissible sets and argumentation stages. In: Computational Dialectics Workshop, pp. 3–7. Citeseer (June 1996)