Knowledge Representation for Argumentation in Agent-Oriented Programming Languages

Alison R. Panisson, and Rafael H. Bordini
Postgraduate Programme in Computer Science — School of Informatics (FACIN)
Pontifical Catholic University of Rio Grande do Sul (PUCRS) – Porto Alegre – Brazil
Emails: alison.panisson@acad.pucrs.br, rafael.bordini@pucrs.br

***Abstract*—Argumentation in multi-agent systems provides both a mechanism for agent reasoning under uncertainty and conflicting information as well as for communication in a more elaborate way, allowing agents to understand each other through the exchange of additional information when compared to other forms of agent communication. Even though argumentation techniques can play an important role in multi-agent systems, little research has been carried out on the issues in integrating argumentation techniques and agent-oriented programming languages, which would allow the development of practical applications taking advantage of such combined techniques. In this work, we present an argumentation framework developed on the basis of an agent-oriented programming language. We cover mainly the practical aspects of such integration, focusing on the knowledge representation expressivity resulting from it. Our approach allows the development of multi-agent applications where agents are able to use arguments in their decision-making processes as well as for communication. The framework has been successfully used as part of the development of a healthcare multi-agent prototype application.**

I. Introduction

Argumentation in multi-agent system, according to [1], can be used for (i) agents reasoning over incomplete, conflicting, and uncertain information/beliefs, resolving conflicts among different arguments and arriving at consistent and well-supported standpoints [2], [3], [4], [5], and (ii) agent communication, allowing the exchange of additional information in dialogues for negotiation, deliberation, and many other important aspects of multi-agent systems [6], [7], [8]. Furthermore, this exchange of additional (compared to alternative approaches) information allows agents to understand each other and to communicate in a more informed way. It also, frequently, changes the mental attitudes of the agents who receive such extra information. All these characteristics just mentioned are important aspects of argumentation-based approaches to multi-agent systems, because of the inherent uncertainty and lack of information present in such distributed systems [9].

Argumentation-based reasoning has been recently brought to the context of agent-oriented programming languages [10], [4]. The integration of argumentation techniques and multi-agent systems by means of an agent-oriented programming language is an important problem to be addressed, as it allows the development of multi-agent applications, composed by agents able to reason about the acceptability of arguments constructed on the face of uncertainty and conflicting information in order to make decisions and to communicate. We argue that some of the major challenges in the integration of argumentation techniques and agent-oriented programming languages are: (i) the means for knowledge representation containing uncertain and conflicting information, (ii) the adaptation/development of the argumentation-based reasoning mechanism to work with such knowledge representation (inferring and defining the acceptability of such inferences), and (iii) how this all is integrated into already existing agent reasoning mechanism underlying the interpreters of agent programming languages.

In this work, we present an argumentation framework developed on the basis of an agent-oriented programming language, focusing in the practical aspects of the integration. The main contribution of this work is the argumentation framework we have developed and in this paper in particular we address: (i) the knowledge representation language we use in order to specify beliefs representing facts/predicates, *strict* and *defeasible* inference; and (ii) the argumentation-based reasoning mechanism integrated with the agent's reasoning mechanism, and how it uses the knowledge representation in order to infer and define the acceptability of arguments;

II. Background

A. Defeasible Logic

Defeasible logic [11] is a particular formalisation of defeasible reasoning, which is a simple and efficient approach to nonmonotonic reasoning. The idea is to formalise nonmonotonic inferences of the type "birds generally fly". Such inferences hold only if there are no inferences to contrary information. Knowledge in defeasible logic is organised as facts, rules that are separated into strict rules and defeasible rules, and a "superiority" relation between rules. Therefore, conclusions can be inferred *strictly* or *defeasibly*. As defeasible rules represent disputable knowledge, defeasible inferences can be defeated by contrary evidence (provided by other inferences).

Defeasible reasoning/logic has a strong connection to argumentation systems [12], specially given the work presented in [13], where this link is established through giving a Dung-like argumentation semantics for defeasible logic [14]. The argumentation semantics proposed is for classical defeasible logic [11] and provides an ambiguity-blocking argumentation system.

B. Agent-Oriented Programming Languages

Among the many agent-oriented programming languages and platforms, such as Jason, Jadex, Jack, AgentFactory, 2APL, GOAL, Golog, and MetateM, as discussed in [15], we chose the Jason platform [16] for our work. Jason extends AgentSpeak(L), an abstract logic-based agent-oriented programming language introduced by Rao [17], which is one of the best-known languages inspired by the BDI (*Beliefs-Desires-Intentions*) architecture, one of the most studied architectures for cognitive agents. In Jason, the agents are equipped with a library of pre-compiled plans that have the following components: (i) a *goal* — the post-condition of the plan (the thing that it achieves); (ii) a *context* — the pre-condition for the plan, defining what must be true of the environment in order for the plan to be successful; and (iii) a *body* — i.e, the 'recipe' part of the plan which contains a sequence of actions and sub-goals in order to achieve the goal for which the plan was written. In particular, plans in Jason have the following syntax:

```
triggering_event : context <- body.
```

where the `triggering_event` represents a new agent goal (or belief in case the plan is to be triggered by reaction to perceived changes in the world) to be pursued and has the format `+!goal(Parameter)`, the `context` has preconditions for the plan to be applicable for achieving that goal, and the `body` is a sequence of actions and sub-goals (which trigger others events and the use of other plans) to achieve the goal. Among the various features of the Jason platform [16], it has some features that are particularly interesting for our argumentation framework, for example: strong negation, belief annotations, and speech-act based communication. [16]

III. Argumentation-based Reasoning in Agent-Oriented Programming Languages

In this section, we describe the argumentation-based reasoning approach we developed. We focus on both the formal specification of the argumentation framework as well as the knowledge representation and argumentation-based reasoning implementation on an agent-oriented programming language. Our framework is constructed on top of the Jason platform [16], and also based on the defeasible logic formalism [11] and its practical implementation called defeasible-Prolog (d-Prolog for short) [18].

In order to present the argumentation-based reasoning mechanism, we divide this section into two parts. First we describe the way we represent defeasible logic [11] and defeasible-Prolog [18] in AgentSpeak (in particular the AgentSpeak dialect available in the Jason platform [16]), including the adaptation of representation and additional possibilities for knowledge representation available in the Jason platform. Second, we describe the argumentation-based reasoning mechanism we developed in order to work with the new knowledge representation internally to the agent reasoning.

A. Defeasible Knowledge Representation

Defeasible logic [11] and defeasible-Prolog [18] have a particular representation in the form of facts, rules, and a "superiority" relation. In order to adequately represent this in AgentSpeak, we represent facts, inference rules, and the superiority relation as belief predicates that are treated similarly to other information in the agent's belief base. As we are working with BDI agents, we use a belief base rather than a knowledge base, assuming that all the agent's knowledge is stored as beliefs in its belief base. So the knowledge representation in our framework is organised as follow: (i) **facts:** are represented as predicates, so for example "Bob is a graduate student" is represented as a simple predicate such as `grad_student(bob)`; (ii) **strict rules:** are represented as a special predicate `strict_inf(Head,Body)`, so for example "graduate students are students" is represented as `strict_inf(student(X),grad_student(X))`; the body can also be a list of predicates to represent conjunction; (iii) **defeasible rules**: are represented as a special predicate `def_inf(Head,Body)`, so for example "graduate students usually study hard" is represented as `def_inf(studies_hard(X),grad_student(X))`; as above, the body can also be a list rather than a single predicate; and (iv) **superiority relation:** the superiority relation is represented as `sup(`$\texttt{Rule}_1$`,`$\texttt{Rule}_2$`)` stating that $\texttt{Rule}_1$ is superior to $\texttt{Rule}_2$. Further, in order to maintain the coherence when we construct inference rules using first-order predicates, as is our case, we allow formulæ such as `X\==Y` into body of rules (i.e., the term instantiated to `X` is different from the one with which `Y` is instantiated), `X==Y` (i.e., equality), and `X>Y` etc. into the list, allowing us to constrain the instantiated variables (within the same predicate or not).

In argumentation-based reasoning, as well as in any argumentation system, an important issue related to knowledge representation is the representation of conflicting information. Regarding conflicting information, we can consider different types of opposition, the most common being *negation* where the negation of a proposition is its opposite in a strong sense of opposition — they are in direct contradiction, for example, "This pen is black" and "This pen is not black". Besides, there is also a weaker sense of opposition, called contraries, for example, "This pen is black" and "This pen is green" [19]. In our argumentation framework, we consider both types of opposition. The strong type, *negation*, is represented using the usual symbol "¬", so "Bob is not a graduate student" is represented as `¬grad_student(bob)`. While the weak type of opposition is represented using a special predicate `comp(good,bad)`, meaning that "good" is contrary to "bad". Therefore, considering the two types of opposition, the notion of conflicting information in our framework can be defined.

Definition 1 (Conflicting Information): Two pieces of information φ and ψ are said to be in conflict if $\varphi \equiv \neg\psi$ or there is `comp(`φ,ψ`)` fact in the belief base. Both cases are represented in our formalisations using a general operator for contradictory information, being $\overline{\varphi}$ contradictory to φ.

Having described how we represent individual pieces of information in our framework, we are now able to describe how arguments are created on top of those. Although the construction of arguments is related to the argumentation-based reasoning mechanism that we will describe in the next section, we describe here the representation of arguments themselves. It is important to note that our argumentation-based reasoning mechanism constructs an argument looking for the inference rules and facts available in the agent's belief base at that particular time in order to derive a specific conclusion from them. Therefore, arguments are composed of a set beliefs — representing the facts and inference rules that can be used as support — and that particular conclusion supported/derived from that set of beliefs.

Definition 2 (Argument): An argument is a tuple $\langle S, c \rangle$, where S is a set of beliefs representing facts and inference rules which supports a conclusion c.

For example, the classical defeasible argument called *argument from perception*, introduced by Pollock [20], says that:

> *"When an object looks red, then (normally, but subject to exceptions) it is red, and this object looks red to me, therefore this object is red."*

This argument, concluding that some particular object is red, is represented as $\langle S, \texttt{color(obj, red)} \rangle$ where S is the set of beliefs representing facts/predicates and inferences rules used to draw that conclusion, as follows:

```
[def_inf(color(obj,red), looks(obj,red)),
                         looks(obj,red)]
```

meaning that the agent believes, through its sensing capabilities, that the object looks red — it has a belief `looks(obj,red)` in its belief base — and the agent has a defeasible rule `def_inf(color(X,Y),looks(X,Y))` whereby it can infer that any object/thing that looks like being of some particular color, can be, at least tentatively, inferred to have that particular color, i.e., in that instance the agent can presumably conclude that object to be red (given also that there is no contrary information to conclude so).

B. Argumentation-based Reasoning Mechanism

The argumentation-based reasoning mechanism, as mentioned before, is used by agents to query the acceptability of conclusions (and their supporting arguments) in their belief bases; such queries are executed internally during the reasoning process. When a conclusion is queried internally by an agent, as is usually the case, its reasoning mechanism searches for that information in their belief base, including acceptable inferences through the extensions we developed.

Before we introduce acceptable arguments (i.e., a conclusion supported by acceptable inferences given the state of an agent's belief base) we need to discuss a few issues. Intuitively, in our argumentation framework we have two kinds of arguments, where arguments that use only *strict* rules are stronger than arguments that use any *defeasible* rules: (i) **strict arguments** are formed only by facts and strict rules (i.e., indisputable knowledge). It is assumed that the strict part of any knowledge base (in practice the agent's belief base) is consistent (i.e., contradictions cannot be derived); (ii) **defeasible arguments** are created using at least one defeasible rule (corresponding to the points of weakness of the argument).

An important aspect of this argumentation framework is that the strict part of the knowledge in the belief base of agents is assumed to be consistent. This is because the strict knowledge is composed by beliefs that can be determined as factual and strict inference rules which are all indisputable knowledge (i.e., they correspond to an agent's knowledge rather than its beliefs), therefore strict knowledge can be naturally assumed to be consistent.

Definition 3 (Consistency of Strict Knowledge): Let Δ^{strict} be the strict part of an agent's belief base Δ. A belief base is said to be consistent if there is no contradictory information derivable from Δ^{strict}, i.e., $\Delta^{strict} \models \varphi$ and $\Delta^{strict} \models \overline{\varphi}$ do not hold simultaneously at any given time.

Whenever an agent queries the acceptability of a particular conclusion, the internal argumentation-based reasoning mechanism developed (extending the normal agent reasoning mechanism) tries to find a rule for that particular queried conclusion and then it attempts to prove the premises of that rule as usual in backwards chaining. In summary, the reasoning mechanism tries recursively to find a prove for that particular conclusion from the beliefs in the belief base, including rules, facts, and assumptions.

In this process, the reasoning mechanism considers conflicting arguments (inferences) in runtime, therefore, in order to define the acceptability of an argument (and the respective queried conclusion) we need to consider conflicting arguments. Conflict between arguments are of two kinds:

Definition 4 (Attack Between Arguments): Let $\langle S_1, c_1 \rangle$ and $\langle S_2, c_2 \rangle$ be two arguments. Attacks between arguments can be generalised into two types:

- The argument $\langle S_1, c_1 \rangle$ rebuts the argument $\langle S_2, c_2 \rangle$ iff $c_1 \equiv \overline{c_2}$.
- The argument $\langle S_1, c_1 \rangle$ undercuts the argument $\langle S_2, c_2 \rangle$ iff $c_1 \equiv \overline{c_3}$ for some $\langle S_3, c_3 \rangle$, where $S_3 \subseteq S_2$.

When two arguments are in conflict, i.e., the arguments attack each other, this does not necessarily mean that one argument defeats the other. Defeat is a "successful" attack, and it considers the set of arguments that defend each other, including preferences between the conflicting arguments [19]. In our framework, the set of acceptable arguments from an agent's belief base is defined in terms of the *defeasible semantics* introduced in [13]. The defeasible semantics is similar to the *grounded semantics* from Dung's work [14], and it is based on the so-called *preempting defeaters* [18]. The preempting defeaters of [18] are called *ambiguity blocking* (in regards to the argumentation system) in [13]. This means that defeasible arguments that are rebutted by as strong as, or stronger, arguments (defined through some sort of preference that will be explained below) are no longer available to rebut other arguments. An example of preempting defeaters is the knowledge base represented by Δ below, where we use $\Rightarrow$ to

refer to *defeasible* inferences:

$$\Delta = \left\{ \begin{array}{lll} a & a \Rightarrow b & b \Rightarrow c \\ x & x \Rightarrow e & e \Rightarrow \neg c \\ y & y \Rightarrow \neg e & c \Rightarrow d \end{array} \right\}$$

In this example, considering Δ as the belief base of some agent, the agent may conclude d using $\{a,\ a \Rightarrow b,\ b \Rightarrow c,\ c \Rightarrow d\}$ as support. Although there is an argument to $\neg c$ supported by $\{x,\ x \Rightarrow e,\ e \Rightarrow \neg c\}$ which rebuts the sub-argument for d that concludes c (undercutting the first argument), this argument (the argument that derives $\neg c$) is defeated (by undercut) by an argument with support $\{y,\ y \Rightarrow \neg e\}$ which prevents the use of that argument to rebut the argument for d.

Although in the example above we have an acceptable argument for d, the arguments with support $\{y,\ y \Rightarrow \neg e\}$ and $\{x,\ x \Rightarrow e\}$ (which is a sub-argument for c in the example) are in conflict, and the argumentation-based reasoning mechanism[1] is not able to decide which one is acceptable, i.e., both are treated as unacceptable. A way to deal with undecided conflicts is to use preferences over the arguments.

Clearly, strict arguments are stronger than defeasible arguments and they have priority, i.e., when arguments are in conflict, strict arguments always defeat defeasible ones. Considering only defeasible arguments, in our framework we have two types of priority: (i) priority by specificity, which is originally defined in defeasible logic [11], and (ii) the explicit declaration of priority between defeasible rules, using a special predicate in the knowledge representation of our argumentation framework. In priority by specificity, more specific conclusions have priority over more general ones. To exemplify this idea, consider the well-known Tweety example:

```
def_inf(flies(X),bird(X)).
def_inf(¬flies(X),penguin(X)).
def_inf(bird(X),penguin(X)).
penguin(tweety).
```

All clauses in the example are defeasible rules. Considering the knowledge above, we have two conflicting arguments, one supporting that Tweety flies: "Tweety flies, because it is a penguin, penguins are birds, and birds fly", and one supporting that Tweety does not fly: "Tweety does not fly, because it is a penguin and penguins do not fly". Our argumentation-based reasoning mechanism (as well as defeasible-Prolog [18] on which our reasoning mechanism is based) concludes, in this case, that Tweety does not fly, because the rule for penguins `def_inf(¬flies(X),penguin(X))` is more specific than a rule for birds `def_inf(flies(X),bird(X))`, given that penguin is a subclass of birds, represented by `def_inf(bird(X),penguin(X))`. In this manner, the argument for Tweety does not fly, `¬flies(tweety)`, has priority over the other one and so defeats it.

Furthermore, when there exist two rules deriving contradictory information, the language used in our approach, as described before, allows us to declare, in an explicit way, priority between these rules, using a special predicate `sup(Rule1,Rule2)`, indicating that inferences using `Rule1` have priority over inferences using `Rule2`. Therefore, when two conflicting arguments are constructed using these conflicting rules, this declaration is used in order to decide which conclusion will actually be derived. Therefore, we can define the acceptability of an argument as follows (based on [13]):

Definition 5 (Acceptable Arguments): An argument $\langle S, c\rangle$ is *acceptable* to an agent *ag* (where Δ_{ag} is its belief base) if $\langle S, c\rangle$ is finite, and: (a) $\langle S, c\rangle$ is strictly inferred, or (b) every argument attacking $\langle S, c\rangle$ is defeated by some argument $\langle S_n, c_n\rangle \in \Delta_{ag}$ (i.e., all arguments that attack $\langle S, c\rangle$ cannot be inferred from Δ_{ag} because they are attacked by as strong as, or stronger, arguments in Δ_{ag} so they are not acceptable in Δ_{ag}).

The reasoning mechanism extending the usual agent reasoning mechanism in Jason in accordance with the argumentation-based reasoning mechanism that we proposed was implemented by adapting d-Prolog [18], using the Prolog-like rules which are interpreted by Jason with some limitations (e.g., the "cut" operator is not available). The implementation is based on logic programming and the formal semantic and syntax of the AgentSpeak language extension that can be found in [16]. A part of that reasoning mechanism is presented below:

```
strict_der(Content):- Content.
strict_der([Content]):- strict_der(Content).
strict_der([First|Rest]):- strict_der(First)
                         & strict_der(Rest).
strict_der(Content):- strict_inf(Content,Cond)
                         & strict_der(Cond).
```

In this part of the implementation we demonstrate the derivation of *strict inferences*, where first we check if the queried content is a belief or a formula (i.e., `X\==Y`, `X==Y`, etc.), then if it is a list of a single element, then if it is a list of more than one element, and finally if it is the `Head` of a strict rule and if the `Condition` (which derives the `Content`) is also strictly derived. These rules permit the agent to query if a content is strictly derived in its knowledge base (remember that strict knowledge is indisputably known)[2].

Therefore, when an agent wants to query if it has an argument to support some conclusion, it uses the special predicates `strict_der(Content)` and `def_der(Content)` depending on whether the agent needs that information to be strictly or defeasibly inferred, respectively. When an agent needs the argument (set of beliefs representing facts and inference rules used in that acceptable inference) to support a claim in a dialogue or a decision-making process, the argument is accessible using a second parameter in the query, which we call `Arg`. Each rule and fact used in the inference of that particular query are stored using the internal action `.concat` (which concatenates a list with the new element, e.g., a rule or a fact) available in Jason. Thus, depending on the strategy of the agent, it can verify if it has a strict or defeasible argument, using `strict_der(Arg,Content)`

[1]This characteristic is from the original implementation of defeasible Prolog [18], and it is what gave rise for the name *ambiguity blocking* in [13].

[2]We do not show here the defeasible part of the mechanism of inference for the sake of space, but our implementation will be made available open source in due course.

and `def_der(Arg,Content)`, or if this distinction is not necessary, the agent can use the special predicate `argument(Content,Arg)` inferred by:

```
argument(Content,Arg):-
strict_der(Arg,Content) | def_der(Arg,Content).
```

where we check first if the content is inferred in a strict way, and then if the content can be inferred in a defeasible way.

The argumentation-based reasoning mechanism queries the acceptability of arguments at runtime, and given the way the agent reasoning cycle works, the arguments are constructed using the most up-to-date information available to the agent given a snapshot of its belief base. Therefore, the acceptability of that particular argument (supporting the queried conclusion) is guaranteed to be in accordance with the updated information available for that agent at the moment of the query. Of course, because of the dynamism of typical multi-agent systems, at the very next reasoning cycle that same argument may no longer be derivable for that same agent.

C. Example

As an example, we adapt the *paper submission scenario* from [4]. In our example, imagine that an agent has submitted a "`paper`" to BRACIS 2016. The agent believes that its paper is good and will be accepted, so it commits itself to buying a ticket to Recife because it concludes `go_to(recife)` from its belief base, given that BRACIS 2016 is to take place in Recife, i.e., it has the belief `held_in(bracis,recife)`.

```
def_inf(go_to(L),[held_in(C,L),accepted(P,C)]).
def_inf(accepted(P,C),[submitted(P,C),good(P)]).
submitted(paper,bracis).
good(paper).
held_in(bracis,recife).
```

A plan that the agent could use to buy a ticket for some location instantiated by `L` has the following format (in the Jason platform):

```
+!buyTicket(L) : def_der(go_to(L))
           <-  buyTicket(L).
```

meaning that if the agent has reasons to believe that it needs to go to some location `L`, it may buy a ticket to that location. When the agent needs the actual argument that supports a particular decision, it could use `argument(go_to(L),Arg)`, thereby instantiating `Arg` with such argument, which could then be used to justify such decision-making to other agents, for example.

However, before the agent buys its ticket, it checks the BRACIS 2016 webpage and realises that the page limit for BRACIS papers is 6 pages (including references) and the agent has, unfortunately, submitted a longer paper than allowed. In addition to the allowed paper length, the agent has the information that papers longer than allowed are *strictly* not accepted. This knowledge is represented as follows:

```
strict_inf(¬accepted(P,C),[longer_for(P,C),
    submitted(P,C)]).
strict_inf(longer_for(P,C),[paper_length(P,X),
    allowed_length(C,Y),X>Y]).
paper_length(paper,9).
allowed_length(bracis,6).
```

With the new information the agent can no longer conclude `go_to(recife)`, considering that the new information allows the inference of a strict argument for `¬accepted(paper,bracis)`, which defeats the argument for `go_to(recife)`, i.e., the argument for `¬accepted(paper,bracis)` successfully undercuts the argument for `go_to(recife)`, considering that they are in conflict and strict arguments have priority over defeasible ones. Therefore, the plan above is no longer *applicable* (the plan's context is no longer satisfied).

IV. Related Work

Much work on reasoning mechanisms based on argumentation can be found in the literature [21], [2], [3], [22], [5], most of them based on abstract argumentation systems at a theoretical level only. Recently, Berariu [10] presented an approach for defeasible reasoning to implement argumentation-based reasoning in BDI agents. The author argues that, given the scarcity of practical work in the area of argumentation-based reasoning, it is now time to address the challenge of putting such well-structured abstract theory into practice, proving its usefulness in real applications. Towards that direction, the system developed in [10] extends the Jason platform with a module for argumentation, which is decoupled from the BDI reasoning cycle, operating in a customised belief base for the agent. Differently from [10], we implement our approach internally to the agents' reasoning mechanism, which we argue is more adequate, considering that in [10], as the mechanism is a decoupled module, the approach works with addition and reaction to extraneous beliefs created to connect that separate module with the agent reasoning, which makes programming agents more difficult and cumbersome, as it is necessary to predict reactions to such belief additions (as responses to the results of the reasoning module).

Another related work is [23], where the authors demonstrate, in a preliminary version, the benefits of argumentation techniques in negotiation between agents. The work is based on the assumption-based argumentation systems [24], and presents a scenario of negotiation between agents (one-to-one negotiation), where the agents negotiate resource allocation. The work is implemented in Jade (Java Agent DEvelopment Framework) [25] and agents engage in dialogues in order to obtain the resources they need. The aim in [23] is to define communication policies which allow agents to provide reasons (arguments) for their refusal to provide the requested resources. The benefits, as the authors describe, are assessed in an informal and experimental way, showing that agents are more effective in identifying the reallocation of resources (or if such reallocations do not exist). Our work differs from [23] because our argumentation framework can be used to develop any type of multi-agent application. Still, as described before our knowledge representation also allows assumptions (as in [23]) and is thus able to implement such communication policies as well.

To the best of our knowledge, [10] and our previous work [4] are the only ones that integrate argumentation in

an agent-oriented programming language, aiming at general argumentation-based reasoning, i.e., where agents are able to reason using the domain-specific knowledge available to them. Therefore, the work presented here is one of the first approaches to integrate argumentation and agent-oriented programming languages in such depth. Further, our argumentation framework is the first that discusses together: (i) argumentation-based reasoning, and defeasible knowledge representation within the context of an agent-oriented programming language; and (ii) how the argumentation-based reasoning mechanism and the knowledge representation are used by agents to construct arguments to support their decision-making and communication/interaction with other agents.

V. Conclusion

In this work, we described an argumentation framework developed on the basis of an agent-oriented programming language. The framework is both formally defined and implemented. Furthermore, in this paper we described the knowledge representation mechanisms available in our framework (i.e., some of the *language* features), which can be interpreted by agents in the same way they have access to their beliefs. Also, we described an argumentation-based reasoning mechanism that allows the agents to construct and define the acceptability of arguments, using both *strict* and *defeasible* inference, meaning that conclusions are drawn in a tentative way and thus might have to be retracted when new information come in. That is, this is an approach to non-monotonic reasoning, where new information can invalidate conclusions previously derived from the agent's belief base.

The argumentation framework presented in this paper allows the development of multi-agent applications that can take advantage of argumentation techniques, both in order to support the agents' decision-making and to allow richer dialogues through the exchange of arguments. In fact, our argumentation framework has been successfully used as part of the development of a healthcare multi-agent prototype application, briefly reported in [6]. In the application, agents are able to argue about task reallocation on behalf of the human users they represent, justifying the requests for reallocation, the refusals to accept particular reallocations, and so on. Further, this argumentation framework has supported the development of the argumentation-based protocol reported in [26].

Our argumentation framework has a working implementation for Jason agents; this implementation and a tutorial with some examples will be made available in the Jason web site[3] in due course. Although we have developed the argumentation framework on the basis of Jason [16], we argue that other agent-oriented programming languages that have similar knowledge representation and inference mechanisms can use our argumentation framework in order to incorporate argumentation techniques.

Acknowledgements

This research was partially funded by CNPq and CAPES.

[3]http://jason.sourceforge.net/

References

[1] N. Maudet, S. Parsons, and I. Rahwan, "Argumentation in multi-agent systems: Context and recent developments." in *ArgMAS*, LNCS, Springer, 2006, pp. 1–16.

[2] L. Amgoud and C. Cayrol, "A reasoning model based on the production of acceptable arguments," *Ann. Math. Artif. Intell.*, vol. 34, no. 1-3, pp. 197–215, 2002.

[3] K. Atkinson and T. Bench-Capon, "Practical reasoning as presumptive argumentation using action based alternating transition systems," *Artif. Intell.*, vol. 171, no. 10-15, pp. 855–874, jul 2007.

[4] A. R. Panisson, F. Meneguzzi, R. Vieira, and R. H. Bordini, "An Approach for Argumentation-based Reasoning Using Defeasible Logic in Multi-Agent Programming Languages," in *ArgMAS*, 2014.

[5] I. Rahwan and L. Amgoud, "An argumentation based approach for practical reasoning." in *AAMAS*, H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, Eds. ACM, 2006, pp.347–354.

[6] A. R. Panisson, A. Freitas, D. Schmidt, L. Hilgert, F. Meneguzzi, R. Vieira, and R. H. Bordini, "Arguing About Task Reallocation Using Ontological Information in Multi-Agent Systems," in *ArgMAS*, 2015.

[7] S. Parsons and P. McBurney, "Argumentation-based dialogues for agent co-ordination," *Group Decision and Negotiation*, vol. 12, no. 5, pp. 415–439, 2003.

[8] S. Parsons, M. Wooldridge, and L. Amgoud, "An analysis of formal inter-agent dialogues," in *1st International Conference on Autonomous Agents and Multi-Agent Systems*. ACM Press, 2002, pp. 394–401.

[9] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[10] T. Berariu, "An argumentation framework for bdi agents," in *Intelligent Distributed Computing VII*, Springer, 2014, vol. 511, pp. 343–354.

[11] ——, "Defeasible logic," in *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, 2001, pp. 353–395.

[12] H. Prakken and G. Vreeswijk, "Logics for defeasible argumentation," in *Handbook of Philosophical Logic, second edition, vol. 4*, D. Gabbay and F. Guenthner, Eds. Dordrecht etc., 2002, pp. 219–318.

[13] G. Governatori, M. J. Maher, G. Antoniou, and D. Billington, "Argumentation semantics for defeasible logic." *J. Log. Comput.*, vol. 14, no. 5, pp. 675–702, 2004.

[14] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial Intelligence*, vol. 77, pp. 321–357, 1995.

[15] R. H. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, *Multi-Agent Programming: Languages, Tools and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[16] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.

[17] A. S. Rao, "AgentSpeak(L): BDI agents speak out in a logical computable language," in *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away*, ser. MAAMAW '96. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996, pp. 42–55.

[18] D. Nute, *Defeasible Prolog*, ser. Research report (University of Georgia. Artificial Intelligence Programs). Artificial Intelligence Programs, University of Georgia, 1993.

[19] D. Walton, C. Reed, and F. Macagno, *Argumentation Schemes*. Cambridge University Press, 2008.

[20] J. L. Pollock, *Cognitive carpentry: A blueprint for how to build a person*. Mit Press, 1995.

[21] L. Amgoud and C. Cayrol, "A reasoning model based on the production of acceptable arguments," in *NMR*, Breckenridge, Colorado, 2000.

[22] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni, "An abstract, argumentation-theoretic approach to default reasoning." *Artif. Intell.*, vol. 93, pp. 63–101, 1997.

[23] A. Hussain and F. Toni, "On the benefits of argumentation for negotiation-preliminary version," in *EUMAS*, 2008.

[24] P. M. Dung, R. A. Kowalski, and F. Toni, "Assumption-based argumentation," in *Argumentation in AI*. Springer, 2009, pp. 199–218.

[25] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, "Jade – a java agent development framework," in *Multi-Agent Programming*. Springer, 2005, pp. 125–147.

[26] A. R. Panisson, F. Meneguzzi, R. Vieira, and R. H. Bordini, "Towards practical argumentation-based dialogues in multi-agent systems," in *IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology*, 2015.