A Modal Defeasible Reasoner of Deontic Logic for the Semantic Web

Efstratios Kontopoulos, Nick Bassiliades, Guido Governatori and Grigoris Antoniou

Author Note

Efstratios Kontopoulos and Nick Bassiliades, Department of Informatics, Aristotle University of Thessaloniki, GR-54124 Thessaloniki, Greece, {skontopo, nbassili}@csd.auth.gr

Guido Governatori, NICTA, Queensland Research Laboratory, PO Box 6020, St Lucia QLD 4067, Australia, guido.governatori@nicta.com.au.

Grigoris Antoniou, Institute of Computer Science, FO.R.T.H., P.O. Box 1385, GR-71110, Heraklion, Greece, antoniou@ics.forth.gr

Correspondence concerning this article should be addressed to:

Nick Bassiliades, Department of Informatics, Aristotle University of Thessaloniki, GR-54124 Thessaloniki, Greece. E-mail: nbassili@csd.auth.gr

Abstract

Defeasible logic is a non-monotonic formalism that deals with incomplete and conflicting information, while modal logic deals with the concepts of necessity and possibility. These types of logics can play a significant role in the emerging Semantic Web, which aims at enriching the available Web information with meaning, leading to better cooperation between end-users and applications. Defeasible and modal logics, in general, and, particularly, deontic logic can assist by providing means for modeling agent communities, where each agent is characterized by its own cognitive profile and normative system, as well as policies, which define privacy requirements, access permissions and individual rights. Towards this direction, this article reports on the extension of DR-DEVICE, a Semantic Web-aware defeasible reasoner, with a mechanism for expressing modal logic operators, while testing the implementation via deontic logic operators, concerned with obligations, permissions and related concepts. The motivation behind this work is to develop a practical defeasible reasoner for the Semantic Web that will take advantage of the expressive power offered by modal logics, accompanied by the flexibility to define diverse agent behaviours. A further incentive is to study the various motivational notions of deontic logic and to discuss the cognitive state of agents as well as the interactions among them.

Introduction

Defeasible Logic, originally introduced by Nute (1994), is a non-monotonic formalism that deals with incomplete and conflicting information. Compared to other more mainstream non-monotonic approaches (Gottlob, 1992; Kautz & Selman, 1991; Soininen & Niemela, 1998), this approach offers enhanced representational capabilities and low computational complexity. Defeasible reasoning can represent facts, rules, priorities and conflicts among rules. Such conflicts usually emerge in the case of rules with exceptions, which are a natural representation for policies and business rules (e.g. work by Rissland & Skalak, 1991 and Schild & Herzog, 1993). In these cases priority information is often used in resolving conflicts among rules.

Modal Logic (Blackburn, de Rijke & Venema, 2001), on the other hand, is a system of formal logic that deals with *modalities*, namely, expressions that are associated with the notions of *possibility* and *necessity*. However, modal logic is not restricted to these concepts, but can assume a variety of interpretations, depending on meaning, context and various other factors. Thus, according to the different interpretations, there exist diverse categories of modal logics, like *epistemic logic* that deals with the certainty of sentences (Meyer, 2001), *deontic logic*, which deals with the notions of obligation and permission (Hilpinen, 2001), *temporal logic*, which deals with temporal notions (Venema, 2001) and *doxastic logic* that deals with reasoning about beliefs (Meyer, 2003).

Modal logics are usually deployed as extensions to classical propositional logic with intentional operators. However, classical propositional logic requires complete, consistent and reliable information, requirements that are rarely met in real-life scenarios, which are by default defeasible in nature. Additionally, reasoning about motivational notions like beliefs, intentions or obligations, displays a significant degree of defeasibility. Consequently, instead of extending propositional logic, it is more appropriate to extend defeasible logic with modal logic elements and recent work confirms this trend (Governatori & Rotolo, 2004; Governatori, Hulstijn, Riveret & Rotolo, A, 2007; Riveret, Rotolo & Governatori, 2007).

The above types of logics are extremely suitable in the *Semantic Web*, which aims at enriching the available information on the Web with meaning, leading to better cooperation between end-users and applications (Berners-Lee, Hendler & Lassila, 2001). Defeasible and modal logics, in general, and deontic logic, in particular, can assist towards this direction, by providing means for modeling *multi-agent systems* (*MAS*), where each agent is characterized by its own cognitive profile and normative system, as well as *policies*, which define privacy requirements for a user, access permissions for a resource, individual rights etc. Already, defeasible logic is applied in a number of related fields, like *semantic brokering* (Antoniou, Skylogiannis,

Bikakis & Bassiliades, 2007), *security policies* (Ashri, Payne, Marvin, Surridge & Taylor, 2004), *e-contracting* (Governatori, 2005) and *agent negotiations* (Skylogiannis, Antoniou, Bassiliades, Governatori & Bikakis, 2007) and, as the article argues, extending defeasible logic with deontic logic operators significantly improves our ability to model applications, naturally using concepts and notions similar to those we find in applications.

This article is based on previous work by the authors (Kontopoulos, Bassiliades, Governatori & Antoniou, 2008) and reports on extending the *DR-DEVICE* first-order defeasible logic reasoner (Bassiliades, Antoniou & Vlahavas, 2006) with reasoning capabilities on modal defeasible logic rule bases. Notice that in this work we focus mainly on deontic logic operators, without loss of generality, even if the techniques proposed can be applied to any modal logic. More specifically, the system has been extended to introduce rule modes that determine the modalities of the derived conclusions, as well as modalized literals that can serve as facts or premises of rule bodies. The aim is to develop a practical defeasible logic reasoner that will take advantage of the expressive power of modal logics, accompanied by the flexibility to define various agent types and behaviors. The motivation behind this work is two-fold, since we wish to: (a) study the various motivational notions, especially those of deontic logic, and (b) discuss the cognitive state of agents as well as the interactions among them, focusing mainly on deontic logic.

The rest of the article is organized as follows: The following section outlines the basics of defeasible logic, focusing mainly on its knowledge representation and inference mechanism, followed by a description of the fundamental notions of modal and deontic logics. The next section summarizes the core notions of extending defeasible logics with modal and deontic logic operators and focuses on two important aspects of modal interactions, namely conflict resolution and rule conversions. The section that follows comprises the backbone of the article and addresses the theoretical and practical issues of extending the DR-DEVICE defeasible logic reasoner with modal and deontic operators, accompanied by a complete use case scenario in the Semantic Web. In our discussion of the implementation we include a proof of the correctness and completeness of the transformation algorithm from a modalized defeasible rule base to an non-modalized equivalent one, along with a discussion on the efficiency of the implementation, which is proved to be in the same order of magnitude in complexity with that of the underlying first-order defeasible reasoning system. Finally, an outline of related ongoing work in the area is presented, as well as concluding remarks and directions for future improvements.

Defeasible Logic

A *defeasible theory D* (i.e. a knowledge base or a program in defeasible logic) consists of three basic elements: a set of *facts* (*F*), a set of *rules* (*R*) and a *superiority relationship* (>). Therefore, *D* can be represented by the triple (*F*, *R*, >). We assume a function-free first-order language.

In defeasible logic, there are three distinct types of rules: strict rules, defeasible rules and defeaters. *Strict rules* are interpreted in the typical sense: whenever the premises are indisputable, then so is the conclusion. Strict rules are denoted by $A \rightarrow p$, where *A* is a set of literals and *p* is a (positive or negative) literal. An example of a strict rule is: "*E-books are books*", which can be formalised as: r_s: ebook(X) → book(X).

Contrary to strict rules, *defeasible rules* can be defeated by contrary evidence and are denoted by $A \Rightarrow p$. Two examples are: r_{d1}: book(X) ⇒ printed(X) ("*Books are typically printed*") and r_{d2}: ebook(X) ⇒ ¬printed(X) ("*Ebooks are not printed*").

Defeaters, denoted by $A \rightsquigarrow p$, are rules that do not actively support conclusions, but can only defeat conflicting defeasible conclusions, by producing evidence to the contrary. An example of a defeater is: r_f: cheap(X) ⇝ ¬printed(X), which reads as: "*Cheap books might not be printed*". This defeater can defeat, for example, rule r_{d1} mentioned above, eventually preventing the derivation of its conclusion.

Finally, the *superiority relationship* among the rule set *R* is an acyclic relation > on *R*, used to resolve conflicts among rules. For example, given defeasible rules r_{d1} and r_{d2}, no conclusive decision can be made about whether an e-book is eventually printed or not, because rules r_{d1} and r_{d2} contradict each other. But, if the superiority relationship $r_{d2} > r_{d1}$ is introduced, then r_{d2} overrides r_{d1} and we can indeed conclude that e-books are not printed. In this case rule r_{d2} is called *superior* to r_{d1} and r_{d1} *inferior* to r_{d2}.

A *conclusion* of *D* is a tagged literal and can have one of the following forms:

- $+\Delta q$, i.e. *q* is definitely provable in *D* (i.e. using only facts and strict rules).
- $-\Delta q$, i.e. we have proved that *q* is not definitely provable in *D*.
- $+\partial q$, i.e. *q* is defeasibly provable in *D*.
- $-\partial q$, i.e. we have proved that *q* is not defeasibly provable in *D*.

If *q* can be proved definitely, then *q* is also defeasibly provable.

Modal and Deontic Logic

Modal logic deals with the notions of *possibility* ("*it is possible that*") and *necessity* ("*it is necessary that*"). In other words, it not only considers truth and falsity, as normal logic does, but also considers what it would be like, if things were different in a possible alternative world. In this sense, something is *necessary* in a world, if it can be proved in all alternative worlds, and it is *possible* in a world, if it can be proved in at least one of all alternative worlds.

Modal logic extends classical logic (either propositional or predicate calculus) by adding two monadic operators that introduce the corresponding modal notions:

- The *necessity operator*, written as □: □*p* ("necessarily *p*")
- The *possibility operator*, written as ◊: ◊*p* ("possibly *p*")

The interpretation of these operators often covers a wider range of concepts with similar rules and a variety of different symbols that can handle an array of other ideas. Thus, modal logic has been applied as the means for establishing the foundations of analyzing epistemic and doxastic notions, like knowledge and belief, paving the way for agents and MAS; modal operators are very useful in expressing the internal cognitive states of agents, as well as interactions among different agents.

A member of the family of logics that stem from modal logic is *deontic logic* that is concerned with permissions, obligations and related concepts. The term "deontic" stems from the Greek corresponding term for "duty". Deontic logic introduces three operators: *O* for "*it is obligatory that*", *P* for "*it is permitted that*" and *F* for "*it is forbidden that*". *P* and *F* are defined in terms of *O*:

$$Pq \equiv \sim O \sim q \qquad Fq \equiv O \sim q$$

Deontic logics can also be built upon propositional logic and can assume an elegant Kripke-style semantics, similarly to standard modal logics. The result is known as *Standard Deontic Logic* (often referred to as *SDL*) and can be axiomatized by adding the following axioms to a standard axiomatization of classical propositional logic:

$O(p \rightarrow q) \rightarrow (Op \rightarrow Oq)$ (if it is obligatory to be that *p* implies *q*, then, if *p* is obligatory, then *q* is obligatory)

$Oq \rightarrow Pq$ (if *q* is obligatory, then *q* is permissible)

Deontic logic is heavily used in expressing policies (Bieber & Cuppens, 1993; Kolaczek, 2002), which are rule sets used in defining access permissions, security and privacy requirements, individual rights etc. for government, private sector organizations and groups, and individuals. Especially for the Semantic Web domain,

policies and automated trust negotiation are two highly appealing applications (Bonatti, Duma, Fuchs, Nejdl, Olmedilla, Peer & Shahmehri, 2006; Kagal, Berners-Lee, Connolly & Weitzner, 2006).

Modal Defeasible Logic

First-order logic has also been used to represent deontic notions (Lokhorst, 1996) leading to formal theories that can be implemented in logic programming systems. Our implementation is one of them, incorporating deontic logic theories in a first-order defeasible logic programming system, called DR-DEVICE. The idea behind this combination is to have a modal framework expressive enough to model certain kinds of deontic defeasibility, in particular by taking into account preferences on norms (Artosi et al., 1996).

In the rest of this article the following modal operators are considered:

- *Belief* (*K*): represents the agent's theory of the world.
- *Intention* (*I*): corresponds to the agent's intentions (policies).
- *Obligation* (*O*): captures the agent's obligations.
- *Agency* (*Z*): represents the agent's intentional actions.
- *Permission* (*P*): corresponds to what the agent is permitted to do.

The permission operator was adopted from Antoniou, Dimaresis and Governatori (2009), allowing the representation of (and reasoning with) business rules and policies. Permissions and obligations form the agent's *normative system* and allow studying the interaction between internal and external factors.

As Governatori and Rotolo (2008a) suggest, there are two potential ways of extending defeasible logics with modal logic operators: (1) adopt explicit predicates that represent the effect of modal operators, or (2) introduce new rule types for modal operators. Thus, the rule "*We intend to go to Rome for the summer*" can be represented in the following two forms respectively:

$$r_1\text{: summer} \Rightarrow I\_\text{weGoToRome} \quad (1)$$

$$r_2\text{: summer} \Rightarrow_I \text{weGoToRome} \quad (2)$$

Option 1 might seem preferable, since it imposes no need for introducing new rule types. Nevertheless, explicit predicates need to be accompanied each time by formal semantics. On the other hand, option 2 enriches the expressiveness of the language and permits interactions among modal operators. The main idea behind this option is to have one defeasible consequence relation for each modal operator. Defeasible logic is able to handle different, and somehow incompatible, intuitions of non-monotonic reasoning. Thus this choice allows us (a) to

have modal operators with different properties, and (b) to have a more fine grained control on how a proposition can be derived and can be used to derive other conclusions. This is not possible if one adopts Option 1.

The second difference between the two options is about the interactions among the various modal operators. To illustrate this difference, suppose, for example, that we have two modal operators $\Box_1$ and $\Box_2$, and the interaction between these two is $\{a :\vdash \Box_1 a\} \subseteq \{a :\vdash \Box_2 a\}$, or in other terms that the two modal operators are related by the axiom $\Box_1 a \rightarrow \Box_2 a$. To capture this, according to the first option, for each proposition/predicate a we need to introduce the propositions/predicates $\Box_1\_a$ and $\Box_2\_a$ and the rule $\Box_1\_a \Rightarrow \Box_2\_a$. For the second option, as we shall see in details in the rest of the paper, we can capture the relationship at the logic level. This means that we can define appropriate derivability conditions. In addition, if we do not want the property to hold for all predicates but only for a few instances, we can capture this at the theory level by introducing rules like $\Box_1 a \Rightarrow_{\Box 2} a$ for the specific predicates. The above discussion illustrates the advantages of the second option over the first, and shows also that the second option is more general and conceptually better, therefore, it is seemingly more appropriate to adopt Option 2 to extend defeasible logics with modal logic operators.

A *modalized literal* is represented by a literal accompanied by a modal operator prefix. For example, the application of rule r_2 above would produce the conclusion `I(weGoToRome)`. A *K*-modalized literal belongs to the knowledge of the environment and is equivalent to the same literal without any prefix. Thus, `K(summer)` and `summer` are equivalent. Consequently, rules with *K* as their mode produce un-prefixed conclusions, if their antecedents are derived. Finally, since rules are meant to introduce modalities, modalized literals appear only in the rule body and not in the head.

With the exception of the belief operator *K*, the rest of the modal operators treated by the system are modeled as *non-reflexive*, meaning that, if *X* is a modal operator, *a* does not follow from *X*(*a*): $X(a) \nrightarrow a$. Also, *iterated modalities*, like *I*(*O*(*a*)) or *O*(*I*(*a*)), that feature iterations of modalities, are not considered in this paper, in order to keep the system manageable (in first-order logic) and because their semantics would be too complicated for the casual Semantic Web end-user. The interested reader can refer to the work by Governatori and Rotolo (2008b) for a better insight into how iterated modalities can be handled. Also *modal interaction axioms*, like $Z(O(a)) \rightarrow I(a)$ and $I(O(a)) \rightarrow I(a)$ are not treated in this paper because there is no general consensus among the researchers on how many and which such axioms should be included in a normative

system. However, any modal interaction axiom can be easily handled through the rule conversion technique used in our system that is presented later in the article.

A conclusion in modal defeasible logic is represented with a tagged literal, similarly to the proof theory of defeasible logics (see previous section), but augmented with information regarding the modality, in which each conclusion is (or is not) proved. Thus, such a conclusion can have one of four forms:

- $+\Delta_X q$: q is definitely provable in modality X in the defeasible theory D.
- $-\Delta_X q$: q is not definitely provable in modality X in the defeasible theory D.
- $+\partial_X q$: q is defeasibly provable in modality X in the defeasible theory D.
- $-\partial_X q$: q is not defeasibly provable in modality X in the defeasible theory D.

An example that illustrates all of the above is the well-known "*prisoner's dilemma*" borrowed from game theory (Hofstadter, 1983): "*Two suspects are arrested by the police. The police have insufficient evidence for a conviction, and, having separated both prisoners, visit each of them to offer the same deal: if one testifies against the other and the other remains silent, the betrayer goes free and the silent accomplice receives the full 10-year sentence. If both remain silent, both prisoners are sentenced to only six months in jail for a minor charge. If each betrays the other, they receive a 5-year sentence. However, the "criminal code of honour" designates not to betray fellow criminals. Each prisoner must make the choice of whether betraying the other or remaining silent. The best individual outcome for each prisoner is to confess the crime, while the best mutual outcome is to stay silent.*"

The above can be formalized in defeasible deontic logic as follows (Governatori & Rotolo, 2004):

f_1: committedCrime

f_2: arrested

p_1: committedCrime $\wedge$ arrested $\Rightarrow_Z$ confess

p_2: committedCrime $\wedge$ arrested $\Rightarrow_O$ $\neg$confess

The dilemma represents a typical conflict between an agent's intentions and normative system (obligations and permissions). According to the agent type, various conclusions can be drawn; for example one can conclude $+\partial_Z$*confess* and $-\partial_O$*~confess* if the agent is *deviant* (intentions and intentional actions override obligations). More details regarding the language and inference of modal defeasible logic can be found in work by Governatori and Rotolo (2004; 2008a).

Conflict Resolution

In the modal extension of defeasible logics a rule is attacked by another rule when the two conclusions are complementary and the two rule modes are different. This conflict resolution scheme is more flexible than the superiority relationship encountered in classical defeasible logics; depending on the modes of the attacking and attacked rules, a set of criteria is designated that determines whether a rule (and which one) eventually prevails. Interestingly, there exist *basic attacks*, which always apply, as well as attacks that depend on the agent type.

Table 1. Basic attacks and corresponding results

$\Rightarrow_K p / \Rightarrow_O \sim p$	$+\partial_K p / -\partial_O \sim p$	$\Rightarrow_I p / \Rightarrow_P \sim p$	$+\partial_I p / +\partial_P \sim p$
$\Rightarrow_K p / \Rightarrow_I \sim p$	$+\partial_K p / -\partial_I \sim p$	$\Rightarrow_P p / \Rightarrow_Z \sim p$	$+\partial_P p / +\partial_Z \sim p$
$\Rightarrow_K p / \Rightarrow_Z \sim p$	$-\partial_K p / -\partial_Z \sim p$	$\Rightarrow_P p / \Rightarrow_O \sim p$	$-\partial_P p / -\partial_O \sim p$
$\Rightarrow_K p / \Rightarrow_P \sim p$	$+\partial_K p / -\partial_P \sim p$	$\Rightarrow_O p / \Rightarrow_I \sim p$	*type of agent*
$\Rightarrow_I p / \Rightarrow_Z \sim p$	$-\partial_I p / -\partial_Z \sim p$	$\Rightarrow_O p / \Rightarrow_Z \sim p$	*type of agent*

In general, beliefs override all other modes, with the exception of agency (i.e. intentional actions). Since beliefs represent an agent's knowledge of the world, they are considered rationally superior to its policies and normative system. Actions, on the other hand, can attack beliefs; because they can have a contradicting effect on the agent's knowledge of the world (i.e. actions can alter the status of the world). As Table 1 shows, no conclusion can be made on the truth of either the belief or the intentional action. This means that knowledge is lost when the agent wants to act in order to change something in the world it believes in. In an actual agent implementation after this conclusion was drawn, the agent would have re-observed the world in order to recover its knowledge.

Intention and agency are mutually attacked; this means that they block each other's proof, since agency (action) is intentional by definition. Mutual attacks also exist among obligation and permission. Agents that encompass this "basic attacks" conflict resolution scheme are often called *realistic* (Broersen, Dastani, Hulstijn, Huang & van der Torre, 2001; Governatori & Rotolo, 2004). Table 1 displays the basic attacks and the corresponding results. For each pair of cells, the first one describes the attack; e.g. the top-left cell features a rule in mode "*K*" that is attacked by a rule in mode "*O*". The second cell displays the resulting conclusions; e.g. in

the second top-left cell, the results of the previous attack are $+\partial_K p$ and $-\partial_O \sim p$, meaning that conclusion p is proved defeasibly, while its complement is not defeasibly provable.

Nevertheless, as the table exhibits, the conflicts among intentions, obligations and actions are resolved, depending on the agent type. Thus, agent types are used in establishing conflict resolution schemes, by determining the way that rules of various modes interact with each other. Table 2 displays the way agent types assist in resolving these conflicts.

The table features six common agent types. The two most "extreme" types are the *independent* agent that can freely adopt intentions and actions that disagree with its normative system and the *pragmatic* agent, which does not allow for any derivation at all. The *selfish saint* has conflicting intentions and obligations, but no intentional action can be derived, while the *social sinner* does not proceed to any action contrary to its obligations, but remains with the intentions to the contrary. Finally, for a *social* agent, obligations override all intentions and actions to the contrary, while *deviant* agents represent the opposite, where obligations are overridden by opposing intentions and intentional actions. Note that our implementation currently includes only the latter two agent types, accompanied by the *realistic* type, but the variety can easily be extended, as explained later.

Table 2. Resolving conflicts among obligations/intentions/actions, depending on agent type

$\Rightarrow_O p$ / $\Rightarrow_I \sim p$		$\Rightarrow_O p$ / $\Rightarrow_Z \sim p$		agent type
$+\partial_O p$	$+\partial_I \sim p$	$+\partial_O p$	$+\partial_Z \sim p$	*independent*
$+\partial_O p$	$+\partial_I \sim p$	$+\partial_O p$	$-\partial_Z \sim p$	*selfish saint*
$+\partial_O p$	$-\partial_I \sim p$	$+\partial_O p$	$-\partial_Z \sim p$	*social*
$-\partial_O p$	$+\partial_I \sim p$	$-\partial_O p$	$+\partial_Z \sim p$	*deviant*
$-\partial_O p$	$+\partial_I \sim p$	$-\partial_O p$	$-\partial_Z \sim p$	*social sinner*
$-\partial_O p$	$-\partial_I \sim p$	$-\partial_O p$	$-\partial_Z \sim p$	*pragmatic*

Rule Conversion

There are certain cases where the conclusion does not "inherit" the rule mode, but it adopts a different modality. This feature, called *rule conversion*, allows for various rational side effects to be derived and is required in capturing certain aspects of the rationality of agents. Rule conversions offer the possibility to convert the rule mode, depending on the modalities, in which the corresponding rule premises have been proved. For

example, if we have rule `weGoToRome` $\Rightarrow_K$ `weGoToItaly`, and we intend to go to Rome, then should we conclude $+\partial_K$*weGoToItaly* or $+\partial_I$*weGoToItaly*? The latter seems more intuitive; thus, the rule can be converted into a rule for intentions. Generally speaking, conversions are a means for deriving rational side effects.

In some cases rule conversion is a debatable principle, and for some combination of modalities and applications it might not be appropriate. Governatori and Rotolo (2004; 2008a) and Governatori, Padmanabhan, Rotolo & Sattar (2009) argue that conversion from beliefs to intention allows us to model the notion of intentionality and that this is essential if one wants to model normative reasoning. In some jurisdiction, the concept of (legal) responsibility depends on that on intentionality, that is if one is aware of the consequences of an intended action (and carries it out), then one intends the consequences as well, unless there are some reasons not to do so. This should be contrasted by the infamous dentist example of Cohen and Levesque (Cohen & Levesque, 1990). The example runs as follows: "*If I intend to the dentist's and I know that if I go to the dentist, I will have pain, it should not follow that I intend to have pain.*" Thus, it seems that subscribing to the conversion principle for belief and intentions leads to counterintuitive conclusions. Let us go back to the example, and let us suppose that I go to the dentist to have a (painful) root canal procedure. Now, let us ask, what are the reasons for going to the dentist to have the procedure? If no further information is given, then, it is reasonable to conclude that I go the dentist to suffer some pain, but if the reasons is that I had a bad cavities that causes some very painful abscess, then the reason is to cure my tooth and this implies that I have the intention not to have pain (Governatori & Rotolo, 2008a).

Table 3. Supported rule conversions

X	⇒	Y	agent type	X	⇒	Y	agent type
O	*K*	*O*	*all types*	*O*	*Z*	*O*	*all types*
I	*K*	*I*	*all types*	*I*	*O*	*I*	*social*
Z	*K*	*Z*	*all types*	*Z*	*O*	*Z*	*social*
P	*K*	*P*	*all types*	*O*	*I*	*O*	*deviant*
I	*Z*	*I*	*all types*	*O*	*Z*	*O*	*deviant*

Conflicts and conversions are not directly related to each other, but they both help portray the cognitive profile of an agent. Nevertheless, similarly to conflicts, there exist rule conversions that apply to all agent types, but there also exist type-dependant conversions.

Table 3 displays the rule conversions supported by the extended DR-DEVICE. The first column indicates the permitted modality of the rule premises. Note that a rule can of course have more than one antecedent; all of them, however, have to share the same modality. Currently, rule conversions for rules with more than one modality in their body are not supported, since they can potentially lead to a combinatorial explosion of cases and also increase the computational complexity of reasoning. The second column indicates the rule mode, while the third one displays the resulting modality of the conclusion. The final column indicates the agent types, for which the specific conversion holds. The selection of agent types can, nevertheless, be extended, as described later.

As a final remark, notice that the contents of the tables above (attacks/corresponding results as well as agent types and rule conversions) merely comprise axioms, which are (philosophically) debatable and with subjective applicability. Of course, each user (developer/knowledge engineer etc) can designate different conflict resolution and conversion schemes that better reflect his/her view of the notions at hand.

Implementation

This section describes the extension of DR-DEVICE, which is called *DR-DEVICE*$_M$ and handles modal and deontic logic operators, rule conversions and conflict resolution schemes.

The DR-DEVICE Defeasible Logic Reasoner

DR-DEVICE (Bassiliades et al., 2006) is a defeasible logic reasoner that employs an object-oriented RDF data model, which is different from the established triple-based data model for RDF. The main difference is that the system treats properties as normal encapsulated attributes of resource objects. This way, properties of resources are not scattered across several triples, as in most other RDF inference systems, resulting in increased query performance due to less joins. The most important features of DR-DEVICE are the following:

- It supports multiple rule types of defeasible logic, such as strict rules, defeasible rules, and defeaters.
- It supports two types of negation (strong, negation-as-failure) and conflicting (mutually exclusive) literals.
- It supports RuleML (Boley, 2006), a mainstream standardization effort for rules in the Semantic Web.
- It supports direct import from the Web and processing of RDF data and RDF Schema ontologies.
- It supports direct export to the Web of the results (conclusions) of the logic program as an RDF document.

- It is built on-top of a CLIPS-based implementation of deductive rules (Bassiliades & Vlahavas, 2006). The core of the system consists of a translation of defeasible knowledge into a set of deductive rules, including derived and aggregate attributes.

As a result of the above, DR-DEVICE is a powerful declarative system supporting: rules, facts and ontologies; major Semantic Web standards: RDF, RDFS, RuleML; monotonic and non-monotonic rules and reasoning with inconsistencies. More details can be found in (Bassiliades et al., 2006) and (Bassiliades & Vlahavas, 2006).

DR-DEVICE supports two types of syntax for defeasible logic rules: a native CLIPS-like syntax and a RuleML-compatible one, called *DR-RuleML*. While DR-RuleML utilizes as many features of the official RuleML as possible, several of the features of defeasible logic cannot be expressed by the existing RuleML specifications. For this reason the modularization scheme of RuleML was applied and new XML Schema documents (up to v.0.91-compatible) were developed. The performed extensions deal with rule types, superiority relations among rules and conflicting literals, as well as with constraints on predicate arguments and functions.

```
<Implies ruletype="defeasiblerule" superior="rd1">
    <oid> <Ind uri="rd2">rd2</Ind> </oid>
    <head>
        <Neg>
            <Atom>
                <op> <Rel>hardcover</Rel> </op>
                <slot> <Ind>name</Ind> <Var>x</Var> </slot>
            </Atom>
        </Neg>
    </head>
    <body>
        <Atom>
            <op> <Rel uri="books:novel"/> </op>
            <slot> <Ind uri="books:name"/> <Var>x</Var> </slot>
        </Atom>
    </body>
</Implies>
```

Fig. 1. Representing rule r_{d2} in the DR-DEVICE RuleML-compatible syntax

Fig. 1 displays the following rule in DR-RuleML (v. 0.91):

r_{d2}: `novel(X)` $\Rightarrow$ ¬`hardcover(X)` (“*Novels are typically not hard-covered*”)

The names (`Rel` elements) of the operator (`op`) elements of atoms are class names, since atoms actually represent CLIPS objects (Bassiliades et al., 2006). RDF class names used as base classes in the rule condition are referred to through the `uri` attribute of the `Rel` element (e.g. `novel` in the figure), while derived class names are text values of the `Rel` element. Atoms have named arguments (slots), which correspond to object/RDF properties. Since RDF resources are represented as CLIPS objects, atoms in the rule body correspond to queries

over RDF resources of a certain class with certain property values, while atoms in the rule head correspond to templates of materialized derived objects, which are exported as RDF resources at the end of the inference process (Bassiliades et al., 2006; Bassiliades & Vlahavas, 2006).

Rule Language Extensions for Modal Logic

The DR-DEVICE RuleML-like syntax (DR-RuleML) described previously was further extended, in order to embrace essential modal logic elements. The language has to describe rule modes as well as a "modalization" mechanism for expressing the modal operator of each literal.

To support these features, two attributes are introduced: `ruleMode` that is appended to the attribute list of the `Implies` element and `modality` that is attached to the `Atom` element. Both attributes can assume only one of five values: `bel` (belief – *K*), `int` (intention – *I*), `obl` (obligation – *O*), `age` (agency – *Z*) and `per` (permission – *P*). The use of the two attributes is optional; thus, when the attribute is absent, the corresponding rule or atom is assigned `bel` (belief) as its mode or modality, respectively.

```
<RuleML ... agentType="social">
    <Assert>
        <Implies ruletype="defeasiblerule" ruleMode="age">
            <oid> <Ind uri="r1">r1</Ind> </oid>
            <head>
                <Atom modality="bel">
                    <op> <Rel>confess</Rel> </op>
                </Atom>
            </head>
            <body>
                <And>
                    <Atom modality="bel">
                        <op> <Rel>committedCrime</Rel> </op>
                    </Atom>
                    <Atom modality="bel">
                        <op> <Rel>arrested</Rel> </op>
                    </Atom>
                </And>
            </body>
        </Implies>
        ...
    </Assert>
</RuleML>
```

Fig. 2. Extending DR-RuleML with modal capabilities

Finally, the notion of agent type is represented by an `agentType` attribute, attached to the rule base element (element `RuleML`). As already mentioned, the currently permitted agent types are *realistic*, *social* and *deviant*, but the selection can be extended by adding more agent types to the schema. Fig. 2 displays a fragment of a "*social prisoner's*" rule base, detailing on rule p_1 below, which states that the intention of an arrested prisoner is to confess the crime to achieve the best individual outcome for him:

p_1: committedCrime $\wedge$ arrested $\Rightarrow_Z$ confess

A similar approach to the rule language presented here was proposed by Pham, Governatori, Raboczi, Newman and Thakur (2008), which is also an extension to the DR-DEVICE RuleML-like syntax. In essence, the two approaches attack the problem similarly, with the difference being that the approach by Pham et al. gives a way to define the agent types in the rule language, whereas in DR-DEVICE we use "preconfigured" types.

DR-DEVICE$_M$ – Overview

Before proceeding to more specific implementation details regarding DR-DEVICE$_M$, this subsection gives an overview of the system functionality. As displayed in Fig. 3, the system accepts as input the address of a modal defeasible logic rule base (step 1 in the figure), written in DR-RuleML. The rule base is submitted to DR-DEVICE$_M$ (step 2) and the theory transformation (described subsequently) commences. During the process, the system deploys an external parameterized modality interaction schema (step 3) and an equivalent non-modal defeasible logic theory is produced (step 4). The latter is then fed to the core reasoner (step 5).

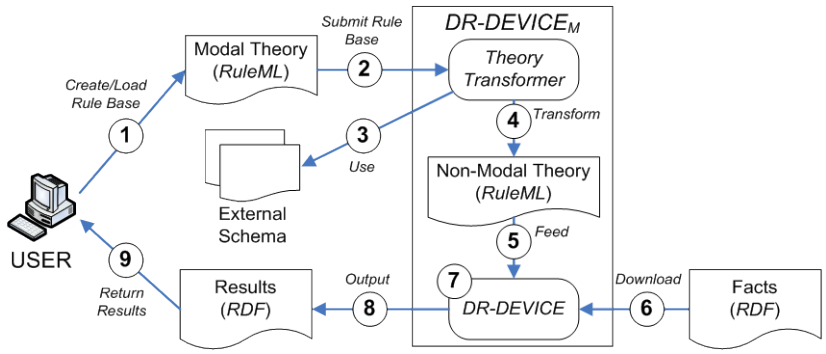

Fig. 3. Modal DR-DEVICE overall functionality.

DR-DEVICE downloads the input RDF documents, including their schemas (step 6) and translates RDF descriptions into CLIPS objects. Rule conclusions are also materialized as objects (step 7) and the instances of designated derived classes are exported as an RDF document (step 8), which includes the RDF Schema definitions for the exported derived classes and those instances of the exported derived classes, which have been proved, either positively or negatively, either defeasibly or definitely. Finally, the user can access the results through a web browser or through specialized software that can customize the visualization (step 9). Notice, that DR-DEVICE can also provide explanations about non-proved objects. More details regarding the functionality and architecture of the system can be found in (Bassiliades et al., 2006).

Theory Transformation

DR-DEVICE$_M$ applies a rule-rewriting transformation to the input modal defeasible logic theory (like the one in Fig. 2) that moves modalities from rules to conclusions and takes care of modal interactions. The transformation is based on the following two binary anti-symmetric relations, introduced subsequently, that define how pairs of modalities interact in *rule conflicts* ($\prec$) and *rule conversions* ($\mapsto$).

Definitions. In both the following definitions, as well as the subsequent sections, $R_Y^X[E(q)]$ will denote the set of rules with mode X, with antecedents that all share the same modality Y and with q as the rule consequent that is modalized under modality E. Note, however, that the above notation is generic and is presented here for representation reasons; it is not possible to concurrently have a rule mode and a modalized conclusion. More specifically, the following distinctions exist:

- When consequent modality E is absent (e.g. $R_Y^X[q]$), then the modality of the consequent is not yet designated.
- When the notation lacks q (e.g. R_Y^X), then the consequent is irrelevant to the expression at hand.
- When index X is absent (e.g. $R_Y[E(q)]$), the rule set consists of modeless rules.
- When index Y is absent (e.g. $R^X[q]$), then the modality of antecedents is irrelevant.

DEFINITION 1 (*rule conflicts*): The relation $\prec$ between two modalities defines how conflicts among conclusions with different modalities are resolved.

$\prec \subseteq M \times M,\ X \in M,\ Y \in M$:

$$Y \prec X \Rightarrow (\forall q\ \forall r\ \forall r',\ r \in R^X[q] \wedge r' \in R^Y[\sim q] \wedge X \neq Y \rightarrow r > r')$$

In the above definition, M is the set of all modality types (in our current implementation $M \equiv \{K, I, O, Z, P\}$) and $R^Y[q] \subseteq R$ denotes the set of rules with mode $Y \in M$ and literal q as their consequent (R is the set of all rules). The definition implies that, for every pair $Y \prec X$, where $X \in M$ and $Y \in M$, if there is an attack $\Rightarrow_X q$ / $\Rightarrow_Y \sim q$, for any literal q in the rule base, the rule with mode X is defined as superior and its conclusion will indeed be derived ($+\partial_X q$). The inferior rule conclusion will not be provable ($-\partial_Y \sim q$), thus, mode X prevails over Y.

The above definition can be extended to consider agent types, as well, meaning that certain conflicts between modalities occur only for specific agent types only and not for others. The relation is augmented with the agent type α:

DEFINITION 1a (*rule conflicts with agent types*):

$\prec_a \subseteq M \times M \times G$, $X \in M$, $Y \in M$, $a \in G$:

$$Y \prec_a X \Rightarrow (\forall r \, \forall r' \, \forall q, r \in R^X[q] \wedge r' \in R^Y[\sim q] \wedge X \neq Y \rightarrow r > r')$$

In the above expression, G is the set of all implemented agent types (currently, $G \equiv \{$`realistic`, `deviant`, `social`$\}$).

Each instantiation of the conflict relation $A \prec_a B$, where $A \in M$ and $B \in M$, is represented by a tuple $<a, A, B>$. Note that the absence of index a referring to the agent type (e.g. $\prec$) indicates that the specified conflict belongs to the basic conflicts set and is present in all agent schemes; actually, agents that adopt the basic conflicts scheme are called *realistic* (the realistic type is the most generic agent type).

DEFINITION 2 (*rule conversions*): The relation $\mapsto$ between two modalities defines how a rule mode is converted into a different conclusion modality when the modality of the condition is different than the modality expected in the original rules.

$\mapsto \subseteq M \times M$, $U \in M$, $Y \in M$:

$$U \mapsto Y \Rightarrow (\forall r, r \in R_Y^U \rightarrow \exists r', r' \in R_Y^Y)$$

In the above definition, $R_Y^U \subseteq R$ denotes the set of all rules with mode $U \in M$ and with $Y \in M$ as the modality of all rule antecedents.

Definition 2 implies that, for every pair $Y \mapsto X$, where $X \in M$ and $Y \in M$, if a rule r exists with mode U and Y as the modality of all antecedents, then a new rule r' is added to the rule base with the same mode as the antecedents of rule r. Notice that relation $\mapsto$ is reflexive, i.e. $\forall X \in M$ it is true that $X \mapsto X$.

The above definition can be extended to consider agent types, as well, meaning that certain modality conversions occur only for specific agent types only and not for others. The relation is augmented with the agent type α:

DEFINITION 2a (*rule conversions with agent types*):

$\mapsto_a \subseteq M \times M \times G$, $U \in M$, $Y \in M$, $a \in G$:

$$U \mapsto_a Y \Rightarrow (\forall r, r \in R_Y^U \rightarrow \exists r', r' \in R_Y^Y)$$

In the above expression, G is the set of all implemented agent types (currently, $G \equiv$ {`realistic`, `deviant`, `social`}).

Each instantiation of the conversion relation $A \mapsto_a B$, where $A \in M$ and $B \in M$, is represented by a tuple $<a, A, B>$. Similarly to definition 1a, the absence of index a (e.g. $\mapsto$) indicates that the specified conversion belongs to the basic conversions set and is present in all agent schemes (i.e. realistic agents).

Transformation steps. The transformation from a modal defeasible theory D_m to a non-modal defeasible theory D_f is influenced by the object-oriented philosophy, on which the system is built (see previous sections). The process consists of three distinct steps that augment theory D_m, so that the resulting defeasible theory D_f is semantically equivalent. Each step takes as input the theory produced by the previous step and produces as output a modified theory to be fed to the next step. The input of the first step is D_m and the output of the last step is D_f.

Before executing the transformation, an initial step that handles a necessary syntactical transformation is performed: for every rule r that belongs to the rule set R_m of D_m, a rule r^o is generated in the rule set R_o of D_o that has the modality of every rule body atom of r transformed from an XML attribute into an argument of the element `atom` of rule r^o. Since atoms in DR-DEVICE are treated as objects, the modality becomes an object slot (or property) with the same name, which is added to the corresponding object definition. This initial step is defined formally as follows:

$$R_o = \{\, r^o(r) \mid r \in R_m \wedge a_i \in r.body.atom \wedge r^o.ID = r.ID \wedge r^o.head = r.head \wedge$$

$$r^o.body.atom = r.body.atom \cup slot(name(\text{"}modality\text{"}), value(a_i.modality)) \,\}$$

Here, we assume a dot-notation, where the expression *e1.e2* delivers the child element or attribute value *e2* of (parent) element *e1*. Functions *name* and *value* are predefined properties of a *slot* element and refer to the slot name and slot value, respectively.

Fig. 4 displays an example of the above process: the left-hand-side excerpt, expressed in the DR-RuleML modal extension described earlier, is the original rule body (rule p_1 from the prisoner's dilemma example seen previously), while the right-hand-side excerpt shows the output body, where modalities have been transformed from atom attributes into object slots (i.e. core DR-RuleML syntax).

To better understand this transformation, rule p_1 is the following:

p_1: committedCrime $\wedge$ arrested $\Rightarrow_Z$ confess

The literals of body of the rule are all under the knowledge or belief modality K (committedCrime$_K$ $\wedge$ arrested$_K$) and during this initial syntactical transformation the modality is transformed into an extra argument of each atom: committedCrime(K) $\wedge$ arrested(K).

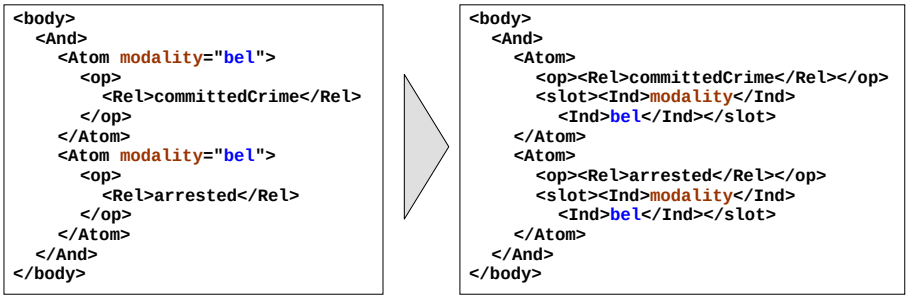

Fig. 4. Example of the initial syntactical transformation.

Then, the actual transformation commences. More specifically, the theory transformation process involves the following steps:

Step 1

This step takes as input the transformed defeasible theory D_o and the corresponding rule set R_o and produces as output the defeasible theory D_1 that contains the rule set R_1. During this step, the mode of each rule (which is an attribute) becomes a slot of the head atom:

$$R_1 = \{ r^1(r^o) \mid r^o \in R_o \wedge r^1.ID = r^o.ID \wedge r^1.body.atom = r^o.body.atom \wedge$$

$$r^1.head.atom = r^o.head.atom \cup slot(name(\text{"}modality\text{"}), value(r^o.mode))\}$$

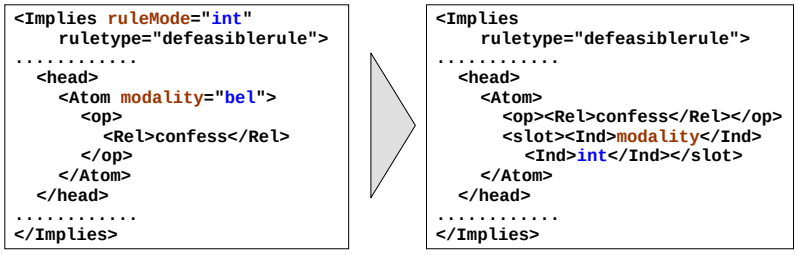

Fig. 5. Example of transformation step 1.

Fig. 5 displays an example of the above process: the left-hand-side excerpt is the original rule head (again, rule p1 from the prisoner's dilemma example), while the right-hand-side excerpt shows the output head, where rule modes have been transferred to head atoms as object slots.

To better understand this transformation step, rule p_1 is the following (after the initial syntactical transformation):

p_1: committedCrime(K) $\wedge$ arrested(K) $\Rightarrow_Z$ confess

After the transformation step 1, the rule looks as follows:

p_1: committedCrime(K) $\wedge$ arrested(K) $\Rightarrow$ confess(Z)

Step 2

This step takes as input the transformed defeasible theory D_1 and the corresponding rule set R_1 and produces as output the defeasible theory D_2 that contains the rule set R_2. During this step, rule conversions are considered and as a result additional rules are added to the rule base. According to rule conversions, the modality of a rule conclusion depends both on the rule mode and on the modalities of the body atoms. So, the rule set R_2 contains all rules of R_1 and the additional rules that take care of rule conversion.

$$R_2 = R_1 \cup \bigcup_{\forall r \in R_1} C(r)$$

The set $C(r)$ contains all additional rules needed for each original rule r. Indeed, many such additional rules could be required (for each rule r), because there can be many qualifying modalities E for conversion.

$$C(r) = \bigcup_{\forall E \in M} \left\{ r' \mid r \in R^X[q] \wedge X \mapsto E \wedge r' \in R_E\left[E(q)\right] \right\}$$

The additional rules $r' \in R_E[E(q)]$ assess whether a rule satisfies the necessary qualifications for rule conversion and are "transparent" to the user, since they are appended during the transformation of the initially submitted theory D_m. Since relation $X \mapsto E$ may hold for several Es for each X, one rule for each E is appended to the theory.

Actually, in order not to clutter the rule base with many rules, instead of appending multiple such conversion rules for each qualifying E, in our implementation we just add one rule whose condition is satisfied for every qualified E. So the previous expression for $C(r)$ is modified as follows:

$$C(r) = \left\{ r' + + X \mapsto E \mid r \in R^X[q] \wedge r' \in R_E\left[E(q)\right] \right\}$$

In the above expression, $r'{+}{+}X{\mapsto}E$ defines a rule r' as previously and concatenates the condition $X{\mapsto}E$ to the rule condition (E is a new universally quantified variable). So, now set $C(r)$ contains just one rule for each r.

Fig. 6 displays an example of a rule added to theory D_f during step 2. The rule (labelled `p1-conv`) is based on rule p_1, but has variable modalities for its body and head atoms in the form of random 4-character strings, like `mt98` and `sw64`. These modalities are checked for their compatibility with the rule mode (function `compatible-modality`) and, if the check is passed, the `convert-modality` function proceeds with the rule conversion (the head modality is converted accordingly).

```
<Implies ruleMode="int" ruletype="defeasiblerule">
  <oid><Ind uri="p1-conv">p1-conv</Ind></oid>
  <head>
    <Atom>
      <op><Rel>confess</Rel></op>
      <slot><Ind>modality</Ind>
      <Var>sw64</Var></slot>
    </Atom>
    <Equal>
      <Var>sw64</Var>
      <Expr>
        <Fun in="yes">convert-modality</Fun>
        <Ind>social</Ind> <Ind>int</Ind>
        <Var>mt98</Var> <Var>yl14</Var>
      </Expr>
    </Equal>
  </head>
  <body>
    <And>
      <Atom>
        <op><Rel>committedCrime</Rel></op>
        <slot><Ind>modality</Ind>
        <Var>mt98</Var></slot>
      </Atom>
      <Atom>
        <op><Rel>arrested</Rel></op>
        <slot><Ind>modality</Ind>
        <Var>yl14</Var></slot>
      </Atom>
      <Equal>
        <Expr>
          <Fun in="yes">compatible-modality</Fun>
          <Ind>social</Ind> <Ind>int</Ind>
          <Var>mt98</Var> <Var>yl14</Var>
        </Expr>
        <Ind>true</Ind>
      </Equal>
    </And>
  </body>
</Implies>
```

Fig. 6. Example of step 2 addition.

To better understand this transformation step, rule p_1 is the following (after transformation step 1):

p_1: committedCrime(K) $\wedge$ arrested(K) $\Rightarrow$ confess(Z)

After the transformation step 2, the following rule is added to the rule base:

p_1-conv: committedCrime(X) $\wedge$ arrested(Y) $\wedge$ compatible-modality(X,Y,Z) $\Rightarrow$

confess(convert-modality(X,Y,Z))

that checks in the condition if *X*, *Y* modalities are compatible with *Z* and if this is true, then they are converted to a compatible with *Z* modality in the rule head using the `convert-modality` function.

Step 3

This step takes as input the transformed defeasible theory D_2 and the corresponding rule set R_2 and produces as output the final defeasible theory D_f that contains the rule set R_f. During this step, the issue of rule attacks (conflicts) has to be tackled with. Typically in DR-DEVICE, two modalized literals (actually objects), such as $I(p)$ and $O(\sim p)$, are not conflicting, since they are treated by the system as different literals/objects (because of the different values in the `modality` slot). In order to allow the reasoner to indeed consider the literals/objects as conflicting, extra rules are added to the rule base, which are also "transparent" to the user. These additional rules realize the so-called *modality inclusion* that deals with the associations among modalities. The results of the attacks are determined by the "basic attacks" scheme (Table 1) as well as the agent type (Table 2), represented by the rule conflicts relation $\prec$. A modality inclusion is denoted by $T_B^A[q] \in T$, where $B \prec A$ (independently of agent type; i.e. the agent type is irrelevant for handling modality inclusions) and T is the modality inclusion set of theory D_f. Each modality inclusion $T_B^A[q]$ results in two defeater rules being added into the rule set R_f: $A(q) \rightsquigarrow B(q)$ and $A(\neg q) \rightsquigarrow B(\neg q)$. Modality inclusions are formulated as defeaters, since they do not aim at deriving new knowledge, but are only used for defeating rules with contrary conclusions. More specifically, for every conclusion q in the rule base, the following modality inclusions have to be incorporated:

$$T_X^Y[q] = \{ Y(q) \rightsquigarrow X(q), Y(\neg q) \rightsquigarrow X(\neg q) \}$$

$$R_f = R_2 \cup \bigcup_{\forall q} \bigcup_{\forall X \forall Y, X \prec Y} T_X^Y[q]$$

Notice that the above definition handles all cases of modality conflicts:

- If $Y \prec X$, i.e. when $+\partial_X q$ / $-\partial_Y \sim q$, then $T_Y^X[q]$ is included in R_f.
- If both $Y \prec X$ and $X \prec Y$ hold, i.e. when $-\partial_X q$ / $-\partial_Y \sim q$, then both $T_Y^X[q]$ and $T_X^Y[q]$ are included in R_f.

As for the remaining two cases of conflicts ($X \prec Y$ and $Y \not\prec X \wedge X \not\prec Y$), the former is the inverse of the first case and is treated similarly, while the latter does not result in the addition of extra rules to the rule base, since both conclusions ($+\partial_X q$, $+\partial_Y \sim q$) are derived.

```
<Implies ruleMode="bel" ruletype="defeater">
  <oid>
    <Ind uri="confess-bel-obl-pos">confess-
    bel-obl-pos</Ind>
  </oid>
  <head>
    <Atom>
      <op><Rel>confess</Rel></op>
      <slot>
        <Ind>modality</Ind>
        <Ind>obl</Ind>
      </slot>
    </Atom>
  </head>
  <body>
    <Atom>
      <op><Rel>confess</Rel></op>
      <slot>
        <Ind>modality</Ind>
        <Ind>bel</Ind>
      </slot>
    </Atom>
  </body>
</Implies>
```

```
<Implies ruleMode="bel" ruletype="defeater">
  <oid>
    <Ind uri="confess-bel-obl-neg">confess-
    bel-obl-neg</Ind>
  </oid>
  <head>
    <Neg>
      <Atom>
        <op><Rel>confess</Rel></op>
        <slot><Ind>modality</Ind>
          <Ind>obl</Ind></slot>
      </Atom>
    </Neg>
  </head>
  <body>
    <Neg>
      <Atom>
        <op><Rel>confess</Rel></op>
        <slot><Ind>modality</Ind>
          <Ind>bel</Ind></slot>
      </Atom>
    </Neg>
  </body>
</Implies>
```

Fig. 7. Example of modality inclusion addition during step 3.

Fig. 7 displays an example of the modality inclusion addition: for conclusion `confess` and for modes K and O, where $O \prec K$, the two following defeaters will be appended to the rule base:

`confess-bel-obl-pos: confess(K)` $\rightsquigarrow$ `confess(O)`

`confess-bel-obl-neg:` $\neg$`confess(K)` $\rightsquigarrow$ $\neg$`confess(O)`

Transformation correctness/completeness. Proving the correctness and completeness of the theory transformation will ensure that the latter always achieves the intended result, i.e. every transformed non-modal defeasible theory is semantically equivalent to the original modal theory, namely, the inferred conclusions are the same. In essence, the aim is to have a transformation from a domain-independent rule base into a domain-dependent theory.

The transformation process comprises an algorithm consisting of three distinct steps plus an initialization step (see previous section). At each step an input rule set is transformed into an output rule set. The input of the initial step is the original, modal rule set R_m, while the output of the last step is an equivalent non-modal rule set R_f. In order to prove that R_f is equivalent to R_m it suffices to prove that each step produces at its output a rule set that is equivalent to its input rule set.

STEP 0:

Input: R_m

Output: R_o

Description: Transforms the original rule set R_m into rule set R_o that has the modality of every rule body atom transformed from an XML attribute into an argument of the atom (XML element).

Proof: This step is a trivial syntactical transformation. Thus $|R_o|=|R_m|$ and there is "1-1" correspondence between rules in the two sets. Furthermore, rules are interpreted similarly. Thus, the two sets are semantically equivalent.

STEP 1:

Input: R_o

Output: R_1

Description: Transforms the rule set R_o into rule set R_1 assigning rule modes of rules in R_o to rule head atom modalities in R_1.

Proof: Rule set R_o contains rules r^o of type $Y(a_1) \wedge Y(a_2) \wedge \ldots \wedge Y(a_n) \Rightarrow_X q$ that all have a mode. Even modeless rules have by default mode *K* (belief). During this step for each rule r^o in R_o one rule rule r^1: $Y(a_1) \wedge Y(a_2) \wedge \ldots \wedge Y(a_n) \Rightarrow X(q)$ is generated in R_1. There exists a "1-1" association among the rule sets of the two theories ($|R_1|=|R_o|=|R_m|$), and, since the corresponding conclusions from the two rules ($+\partial_X q$ and $+\partial X(q)$) are semantically equivalent, step 1 is proved to be correct and complete.

STEP 2:

Input: R_1

Output: R_2

Description: Transforms the rule set R_1 into rule set R_2 by adding new rules that assess, whether a rule qualifies for rule conversion.

Proof: For each rule r^1: $Y(a_1) \wedge Y(a_2) \wedge \ldots \wedge Y(a_n) \Rightarrow X(q)$ in R_1, and for each *E*, such that $X \mapsto E$, a new rule $E(a_1) \wedge E(a_2) \wedge \ldots \wedge E(a_n) \Rightarrow E(q)$ should be appended to the theory. However, in order not to clutter the rule base with many rules, instead of appending multiple such conversion rules for each qualifying *E*, we add just one such rule that its condition is satisfied for every qualified *E*, namely the following rule is added to the rule base for each rule r^1: $E(a_1) \wedge E(a_2) \wedge \ldots \wedge E(a_n) \wedge X \mapsto E \Rightarrow E(q)$. Thus, $|R_2|=2 \cdot |R_1|=2 \cdot |R_m|$. Furthermore, according to formal modal defeasible logics, applying rule conversion on a rule such as $E(a_1) \wedge E(a_2) \wedge \ldots \wedge E(a_n) \Rightarrow_X q$ would result in concluding $+\partial_E q$. Following our approach, the appended rule $E(a_1) \wedge E(a_2) \wedge \ldots \wedge E(a_n) \wedge X \mapsto E$

$\Rightarrow E(q)$ results in $+\partial E(q)$, which is semantically equivalent to $+\partial_E q$, therefore, correctness and completeness of step 2 are proven.

STEP 3:

Input: R_2

Output: R_f

Description: Transforms the rule set R_2 into the final rule set R_f by adding new rules that tackle modality inclusion, namely rule attacks (conflicts).

Proof: During this step a number of defeater rules are added to rule set R_2. For every pair of conflicting heads $X(q)$ and $Y(\neg q)$, where $Y \prec X$, two new defeater rules are appended to the rule set R_2. Thus, if $\alpha \in \mathbb{N}$ is the number of $\prec$ relation instances (actually, the number of modality attacks) occurring in D_m and c is the number of distinct (non-modalized) literals occurring at rule heads in R_m, then $|R_f|=|R_2|+2 \cdot \alpha \cdot c = 2 \cdot |R_m|+2 \cdot \alpha \cdot c$. In the worst case $c=|R_m|/2$, i.e. each conflicting rule pair has a distinct conclusion; therefore $|R_f| = 2 \cdot |R_m|+\alpha \cdot |R_m| = |R_m| \cdot (\alpha+2)$, which is always finite. Also, according to formal modal defeasible logics, if there exists a pair of rule consequents $\Rightarrow_X p$ / $\Rightarrow_Y \sim p$ with $Y \prec X$, then we conclude $+\partial_X p$ and $-\partial_Y \sim p$. Similarly, according to our approach, the addition of modality inclusion $T_Y^X[p]$ will result in defeaters $X(p) \rightsquigarrow Y(p)$ and $X(\neg p) \rightsquigarrow Y(\neg p)$ being added to theory D_f. Consequently, these defeaters result in concluding $-\partial_Y \sim p$ and allowing the eventual inference of $+\partial_X p$. Thus, step 3 is also correct and complete.

Implementing Modality Interactions

As studied previously in the article, each type of agent is accompanied by its own modality interaction schema, namely, a distinct set of conflicts and conversions. In order to represent modality interactions in DR-DEVICE$_M$, in a simple yet expressive way that could facilitate re-use and evolution, a generic XML-Schema-based representation is adopted. The representation allows defining agent types along with the respective sets of conflicts and conversions, with the help of the relations $\prec$ and $\mapsto$, described earlier.

More specifically, for the case of attacks we define the set L of all attacks as:

$$L = T_B \cup \bigcup_{\forall a} T_a$$

where T_B is the set of basic attacks (see Table 1), which is common for all agent types and T_a is the set of attacks that are specific for agent type $a \in A$. Similarly, the set V of conversions is defined as:

$$V = N_B \cup \bigcup_{\forall a} N_a$$

where N_B is the set of basic rule conversions (see Table 3), which is common for all agent types, and N_a is the set of rule conversions that are specific for agent type a. Therefore, for each agent type $a \in A$, its conflict and conversion sets (L_a and V_a respectively) are defined as follows:

$L_a = T_B \cup T_a$, $T_a \equiv \{<a, X, Y> \mid Y \prec_a X\}$

$V_a = N_B \cup N_a$, $N_a \equiv \{<a, Y, U> \mid U \mapsto_a Y\}$

```
<!-- root element -->
<!ELEMENT agents (agent*)>

<!--
Each agent is characterized by a conflict
and a conversion set
-->
<!ELEMENT agent (conflicts?, conversions?)>
<!ATTLIST agent type ID #REQUIRED>

<!-- agent conflict set -->
<!ELEMENT conflicts (conflict*)>

<!--
The 1st mode is the superior rule mode
The 2nd mode is the inferior rule mode
The superior rule always prevails
-->
<!ELEMENT conflict (mode, mode)>

<!-- agent conversion set -->
<!ELEMENT conversions (conversion*)>

<!--
The 1st mode occurs in the body
The 2nd mode is the rule mode
-->
<!ELEMENT conversion (mode, mode)>

<!-- element representing mode/modality -->
<!ELEMENT mode (#PCDATA)>
```

```
<agent type="social">
    <conflicts>
        <conflict>
            <mode>BEL</mode>
            <mode>OBL</mode>
        </conflict>
        <conflict>
            <mode>BEL</mode>
            <mode>INT</mode>
        </conflict>
        <conflict>
            <mode>OBL</mode>
            <mode>INT</mode>
        </conflict>
    </conflicts>
    <conversions>
        <conversion>
            <mode>OBL</mode>
            <mode>BEL</mode>
        </conversion>
        <conversion>
            <mode>INT</mode>
            <mode>BEL</mode>
        </conversion>
        <conversion>
            <mode>OBL</mode>
            <mode>INT</mode>
        </conversion>
    </conversions>
</agent>
```

Fig. 8. Modality interaction schema and excerpt defining the social agent type.

The schema is illustrated as a DTD (for simplicity) in Fig. 8. The figure also depicts a sample XML excerpt that defines the *social* agent type.

Implementation Efficiency

In this subsection we briefly discuss the efficiency of our implementation in reference to the computational complexity of the underlying logic and that of the transformed rule base. We claim that the reasoning computational complexity of the transformed rule base is $O(n^2)$, where n is the number of rules in the original rule base. In order to prove this, we need to have as input:

- The computational complexity of the underlying logic, which is a first-order defeasible reasoning system.
- The computational complexity of the transformed rule base.

Our underlying defeasible reasoning system is DR-DEVICE (Bassiliades et al., 2006). DR-DEVICE has undergone extensive testing with scalable knowledge bases which showed that the system can handle inferences with thousands of rules within few seconds. Its performance scalability is $O(n^2)$, where n is the size of the theory, i.e. the total number of defeasible rules.

As shown in the previous subsection, the transformed rule base (the one with non-modalized literals) contains $m\cdot(\alpha+2)$ rules, where m is the number of rules in the modalized rule base and α is the number of modality attacks (namely $|L_a|$, the cardinality of the L_a set, for each agent type a**)**. Upon scaling, the number of rules tends to grow quite larger than α. For example in (Bassiliades et al., 2006) m scaled up to 10^3, whereas α is usually less than 10 (for example, in our implementation the "social" agent type has 3 modality attacks, see Fig. 8). If we replace this in the complexity of the DR-Device system we can conclude that the complexity of our implementation is $O(m^2\cdot a^2)$, therefore assuming that $m>>\alpha$ the complexity is $O(m^2)$, i.e. it is quadratic to the number of rules of the original modalized rule base, as our initial claim was.

Example: Prisoner's Dilemma

To illustrate all of the above, we use an example that features the prisoner's dilemma rule base. Notice that, although DR-DEVICE$_M$ is a first-order system, this example is in propositional logic for the sake of simplicity for the demonstration. In the next subsection we include a predicate logic example.

Initially, the rule base contains two facts (f_1 and f_2) and two rules (p_1 and p_2):

f_1: committedCrime

f_2: arrested

p_1: committedCrime $\wedge$ arrested $\Rightarrow_Z$ confess

p_2: committedCrime $\wedge$ arrested $\Rightarrow_O$ $\neg$confess

Let's suppose that we deal with a *social* agent. The initial rule base undergoes the transformation described previously, before being submitted to the defeasible logic reasoner of the system. As already outlined, the transformation includes turning predicate modalities into atom slots and assigning modalities to rule heads, depending on the corresponding rule modes (step 1). Predicates with no modality assume modality *K* (knowledge). As a result, at this stage the transformed rule base will be as follows:

$f_{1\text{-CONV}}$: K(committedCrime)

$f_{2\text{-CONV}}$: K(arrested)

$p_{1\text{-CONV}}$: K(committedCrime) $\wedge$ K(arrested) $\Rightarrow$ Z(confess)

$p_{2\text{-CONV}}$: K(committedCrime) $\wedge$ K(arrested) $\Rightarrow$ $\neg$O(confess)

Then, the issue of rule conversion, i.e. the conversion of rule conclusion modality according to the condition modality, is handled by the addition of two rules at step 2:

$p_{1\text{-CONV-VAR}}$: M(committedCrime) $\wedge$ M(arrested) $\wedge$ Z $\mapsto$ M $\Rightarrow$ M(confess)

$p_{2\text{-CONV-VAR}}$: M(committedCrime) $\wedge$ M(arrested) $\wedge$ O $\mapsto$ M $\Rightarrow$ $\neg$M(confess)

The above rules (e.g. $p_{1\text{-CONV-VAR}}$) indicate that if the modality of all condition literals is *M* and *M* is a modality that is compatible with the original rule mode *Z*, then it is indeed possible to derive the conclusion of the rule with modality *M*.

Finally, the issue of rule attacks has to be dealt with. In the rule base there is only a single conclusion (confess) and a single attack ($\Rightarrow_Z$ *confess* / $\Rightarrow_O$ *~confess*). Therefore, two defeaters are added in step 3 for the specific agent type (any obligation is also the agent's intentional action), as seen below:

$p_{O\text{-Z-POS}}$: O(confess) $\rightsquigarrow$ Z(confess)

$p_{O\text{-Z-NEG}}$: $\neg$O(confess) $\rightsquigarrow$ $\neg$Z(confess)

Rule $p_{1\text{-CONV}}$ is defeated by $p_{O\text{-Z-NEG}}$ and it is eventually concluded that $+\partial_O$*~confess* and $-\partial_Z$*confess*, which implies that the social agent's obligation to confess overpowers its intention not to confess.

A slight variation of the example that demonstrates rule conversion would include facts f_1' and f_2' instead of f_1 and f_2:

f_1': Z(committedCrime)

f_2': Z(arrested)

In this case no conclusion is derived, because rule $p_{2\text{-CONV-VAR}}$ tries to conclude ¬Z(confess), since the agent is *social* and O↦Z, while rule $p_{1\text{-CONV-VAR}}$ tries to conclude Z(confess), because Z↦Z. Thus, no conclusion on the confession can be derived, even though the agent is social and this would incline to its cooperativeness. Although the rationality of this last example is dubious, cases like this are useful in studying the "*side-effects problem*" (Governatori & Rotolo, 2008a; Governatori et al., 2009) (i.e. results from agent's actions that are not among its intentions, at least not *all* of them), which is one of the most interesting topics of discussion regarding the *rationality* and *awareness* of agents.

A Semantic Web Example: Apartment Renting

This subsection describes a more sophisticated example, in predicate (first-order) logic, that is oriented towards the Semantic Web paradigm. The example is adopted from Antoniou & van Harmelen (2004) and is applied as a use case scenario for the DR-DEVICE defeasible reasoner (Bassiliades et al., 2006). In its original form the scenario involves two parties: (a) A potential renter, who assigns his/her personal agent to discover an apartment to rent that suits his/her specifications (e.g. location, floor) and personal preferences (e.g. price, size), and (b) a broker, who possesses a list of available apartments and has to match the client's requirements with the features of the available apartments and eventually propose suitable flats.

In the Semantic Web environment, the broker's list of available apartments along with their specifications is materialized as an RDF document (see Fig. 9) that complies with a corresponding RDF Schema ontology[1]. Actually, this piece of data (also known as "fact" in logic programming terminology), is equivalent to the following in POSL notation:

```
apartment(name->a1, bedrooms->1, central->yes, floor->1, gardenSize->0, lift->no,
          pets->yes, price->300, size->50).
```

POSL (positional-slotted language) notation (Boley, 2004) is an ASCII language that integrates Prolog's positional and F-logic's slotted syntaxes for representing knowledge (facts and rules) in the Semantic Web. For representing defeasible rule bases in a concise syntax, we proposed the d-POSL notation in (Kontopoulos, Bassiliades, Governatori, & Antoniou, 2010) and we use this notation to concisely present rules of this example in this paper. When compared to RuleML-based types of syntax (like e.g. DR-RuleML – see Fig. 2), d-POSL is considered more compact and human-readable, since it is faster to write and easier to read. Variables in d-POSL

[1] http://lpis.csd.auth.gr/systems/dr-device/carlo/carlo.rdf

are denoted with a preceding "?", rule types (“strict”, “defeasible”, “defeater”) are expressed via binary infix functors (“:-”, “:=”,“:~”), while the extension also deals with other essential features in defeasible logic, like rule labels, rule superiorities and conflicting literals. More details regarding the application of d-POSL in an agent-based brokering scenario can be found in (Kravari, Kontopoulos & Bassiliades, 2010b).

```
<ex:apartment rdf:about="&ap;a1">
 <ex:name>a1</ex:name>
 <ex:bedrooms rdf:datatype="&xsd;integer">1</ex:bedrooms>
 <ex:central>yes</ex:central>
 <ex:floor rdf:datatype="&xsd;integer">1</ex:floor>
 <ex:gardenSize rdf:datatype="&xsd;integer">0</ex:gardenSize>
 <ex:lift>no</ex:lift>
 <ex:pets>yes</ex:pets>
 <ex:price rdf:datatype="&xsd;integer">300</ex:price>
 <ex:size rdf:datatype="&xsd;integer">50</ex:size>
</ex:apartment>
...
```

Fig. 9. RDF fragment representing available apartments.

```
<RuleML rdf_import="...carlo_ex.rdf" rdf_export="export-carlo.rdf" >
.....
<Implies ruletype = "defeasiblerule">
 <oid><Ind uri = "&carlo_rb;r1">r1</Ind></oid>
 <head>
  <Atom>
   <op><Rel>acceptable</Rel></op>
   <slot><Ind>apartment</Ind><Var>x</Var></slot>
  </Atom>
 </head>
 <body>
  <Atom><op><Rel uri = "carlo:apartment"/></op>
   <slot><Ind>carlo:name</Ind><Var>x</Var></slot>
  </Atom>
 </body>
</Implies>
.....
</rulebase>
```

```
<!DOCTYPE rdf:RDF [
<!ENTITY dr-device "http://.../dr-device/export/export-carlo.rdf#" > ]>
<rdf:RDF... xmlns:dr-device='&dr-device;'>
.....
<dr-device:acceptable rdf:about="&dr-device;acceptable5">
<dr-device:apartment>a5</dr-device:apartment>
<dr-device:truthStatus>defeasibly-proven</dr-device:truthStatus>
</dr-device:acceptable >
.....
</rdf:RDF>
```

Fig. 10. DR-RuleML fragment of the renter’s requirements (left) and DR-DEVICE derived conclusions (right).

Assuming that the schema is publicly available, the renter’s requirements and preferences are based on it as well and are expressed in defeasible logic, in the DR-RuleML syntax (see Fig. 10 – left). The rule r_1 is expressed as follows in d-POSL notation:

```
r1: acceptable(apartment->?x) := apartment(name->?x).
```

The final conclusion regarding the most appropriate apartment for rental is derived via inference by the DR-DEVICE reasoner over the rule set and the RDF list of available apartments (see Fig. 10 – right). Suppose that the final conclusion is `rent(apartment->a5)` (in d-POSL), where `a5` is the id of the most appropriate

apartment. Notice that in the OO-RDF model of DR-DEVICE properties, like `apartment`, are encapsulated inside their domains, such as `rent` class.

The above example can be extended with deontic defeasible logic elements, in order to illustrate the applicability of this type of logic in pragmatic Semantic Web scenarios. According to a "deontic" point of view, since the renter has used a specific broker's services, he/she is "morally obliged" to make an offer for one of the broker's apartments (specifically for apartment `a5`, which is the most appropriate one for him). This obligation is expressed via rule:

r_{M1}: `make_offer(apartment->?x)` $:=_O$ `rent(apartment->?x).`

Continuing the "deontic" version of the apartment renting scenario, let's suppose that the renter has conducted on the Web another market research by himself/herself and has formed a list of advertised apartments, expressed via predicate `advertised_apartment(name->?x,price->?y)`. Let's also suppose that he/she has performed the selection process himself/herself on the latter list (and not on the broker's apartment list), retrieving his/her favorite apartment expressed e.g. as `my_rent(apartment->b2)`. If the renter's agent discovers a cheaper favorite apartment than the one proposed by the broker (apartment `a5`), his/her intention is to *not* make an offer for the broker's apartment (and maybe later make an offer to the apartment he/she found on the web):

r_{M2}: $\neg$`make_offer(apartment->?x)` $:=_I$

`rent(apartment->?x), apartment(name->?x, price->?y),`

`my_rent(apartment->?z), advertised_apartment(name->?z, price->?w),`

`?x` $\neq$ `?z, ?w < ?y.`

Considering rules r_{M1} and r_{M2} above, a deviant agent's intentions prevail over its obligations to the contrary and the agent will eventually avoid making an offer for apartment `a5`. Contrasted, a social agent would prefer to obey to the moral obligation of preferring the broker's suggestion and would eventually make an offer for apartment `a5`.

To demonstrate the effect of rule conversions, an initial precondition of I`(rent(apartment->a5))` would produce via rule r_{M1} (supposing that the renter's agent is *social*) the corresponding rule conversion $+\partial_I$ *make_offer*(*apartment->a5*). Supposing also that the rule's r_{M2} prerequisites hold as well, the second

conclusion would be $+\partial_I \neg$*make_offer*(*apartment->a5*). The two conclusions are conflicting and none of them prevails, if no superiority relationship is explicitly expressed in the rule base.

Related Work

The work presented here relies heavily on work by Governatori and Rotolo (2004; 2008a), where a detailed account on extending defeasible logics with modal logic operators is given. The main differentiation among the two lines of research lies in the fact that the work by Governatori and Rotolo is based on the meta-program formalization presented by Maher and Governatori (1999), while our approach adopts a theory transformation for turning a modal defeasible theory into a non-modal one. Other secondary differences involve the permission operator (which, however, is included in the ideas for future work by the other authors) and the definition of the agent type.

An implementation approach with similar functionality to ours is proposed by Antoniou, Dimaresis and Governatori (2009). The proposed system is based on *DR-Prolog* (Antoniou & Bikakis, 2007), a Prolog-based defeasible reasoner for the Semantic Web. The approach extends the meta-program for simulating modal defeasible logic (Antoniou, Billington, Governatori & Maher, 2006) and the corresponding inference mechanism. Similarly to DR-DEVICE$_M$, the proposed system also deals with rule conflicts and rule conversions and can also describe various agent types. The main difference among the two implementations lies in the superior centralization and modularization offered by our system. The type of the agent can be declared centrally in each rule base, while modality interactions (attacks and conversions) are dealt with parametrically (i.e. via easily-parameterized external agent-type definition files), as described in this work. Also, the RuleML-like language for describing a modal defeasible logic rule base is quite easy to grasp, since the extensions to the language are limited.

Another similar paradigm is *SPINdle* (Lam & Governatori, 2009), a propositional defeasible logic reasoner that covers both the standard and modal extensions to defeasible logics. Regarding efficiency, when compared to DR-DEVICE$_M$, the system demonstrates low computational complexity during reasoning coupled with low memory consumption. In *SPINdle*, agent types can be defined by specifying the conversion and conflict relationships using the rule language, while in DR-DEVICE$_M$ there are predefined agent types in a separate configuration file. This may be more user-friendly and less error-prone than the approach of *SPINdle*, but could be less flexible. Finally, DR-DEVICE$_M$ and the modal extension of DR-Prolog are more RuleML-compatible

and they are more expressible than *SPINdle*, since they are first-order. On the other hand, the modal extension of DR-Prolog is query based while DR-DEVICE$_M$ and *SPINdle* are extension based.

Taking into consideration more traditional approaches, the approach presented in this article has a lot of common points with the *BOID* architecture (Broersen et al., 2001), where the conflicts among agents' informational and motivational attitudes are explored. The BOID calculation scheme is similar to the one proposed here; e.g. it is possible to state general orders of overruling but also local preferences involving single rules. However, our system also deals with agency and permission, a factor that largely differentiates our approach from the BOID architecture, and is also designed to take care of modalized literals and modal conversions. This is due to the rule-based approach of introducing modalities.

Also, Nute (1998) proposed a *Deontic Defeasible Logic* which, in some respect, is similar to the framework presented here. Besides some minor differences in the way rules are handled at the propositional level, the main difference is that he uses only one type of rule. In proof theory operators are traditionally introduced for giving meaning to rules. Thus, using only one type of rule both for obligation and factual conclusion does not reveal the real meaning of the operators involved. Moreover, it is not clear whether and how complex conversions and reductions can be dealt with in a system with only a single type of rule.

Conclusions and Future Work

This article discusses the extension of defeasible logics with modal logic operators and reports on the implementation of a modal defeasible logic reasoner, focusing mainly on deontic logic operators. The implementation is based on DR-DEVICE, a defeasible logic reasoner over RDF metadata in the Semantic Web. The system was extended, so that it can handle modal logic operators, while testing the implementation with deontic logic operators. More specifically, five operators are included: knowledge, intention, obligation, agency and permission. The article also demonstrated how the RuleML-like language of DR-DEVICE, called DR-RuleML, was extended to incorporate the necessary modal logic aspects. Besides modal operators and modalization of literals, the system also deals with modal interactions, like conflict resolution and rule conversion. The submitted defeasible theory undergoes a transformation, imposed by the object-oriented philosophy, on which the system is built, so that the modal defeasible reasoning can be successfully performed.

Nevertheless, there is still plenty of room for improvement. More specifically, the system currently handles only three types of agents, realistic, social and deviant, but the selection can easily be extended to

include other agent types as well, starting with the ones in Table 2. Defining new agent types will consequently allow the enrichment of the available rule conversions, which will lead to a more expressive rule language but also to a more thorough study of the "side-effects problem". Furthermore, we would like to test our implementation by introducing more operators from other modal logics, such as temporal, epistemic, doxastic, etc. Finally, more complicated features of modal logics, such as iterated modalities and modal interaction axioms, will be incorporated in our implementation by extending the existing transformation mechanisms. Overall, the aim of this line of work is to observe and explore the interactions among agents of various cognitive profiles.

A more ambitious goal, though, is to utilize a multi-agent system, such as *EMERALD* (Kravari, Kontopoulos & Bassiliades, 2010a), for deploying argumentation scenarios among agents, each one of which will possess its own agenda and normative system. EMERALD is a knowledge-based framework for interoperating intelligent agents in the Semantic Web. It is built on top of the JADE[2] multi-agent development framework and features trusted, third party reasoning services, a reusable agent prototype for knowledge-customizable agent behavior, as well as a reputation mechanism for ensuring trust in the framework.

[2] http://jade.tilab.com/

References

Antoniou, G., & Bikakis, A. (2007). DR-Prolog: A System for Defeasible Reasoning with Rules and Ontologies on the Semantic Web. *IEEE Transactions on Knowledge and Data Engineering, 19*, 2, 233-245.

Antoniou, G., Billington, D., Governatori, G., & Maher, M. J. (2006). Embedding Defeasible Logic into Logic Programming. *Theory and Practice of Logic Programming, 6*, 6, 703-735.

Antoniou, G., Dimaresis, N., & Governatori, G. (2009). A Modal and Deontic Defeasible Reasoning System for Modelling Policies and Multi-Agent Systems. *Expert Systems With Applications, 36*, 2, 4125-4134.

Antoniou, G., & van Harmelen, F. (2004). *A Semantic Web Primer*. MIT Press, ISBN-10: 0262012421.

Antoniou, G., Skylogiannis, T., Bikakis, A., & Bassiliades, N. (2007). DR-BROKERING: A Semantic Brokering System. *Knowledge-Based Systems, 20*, 1, 61-72.

Artosi, A., Governatori, G., & Sartor, G. (1996). Towards a computational treatment of deontic defeasibility. In Mark Brown and José Carmo, eds. *Deontic Logic Agency and Normative Systems, Workshop on Computing*, pp. 27-46, Springer-Verlag, Berlin.

Ashri R., Payne T., Marvin D., Surridge M., & Taylor S. (2004). Towards a Semantic Web Security Infrastructure. *Proceedings of the AAAI Spring Symposium on Semantic Web Services Semantic*, pp. 84–91, AAAI Press, Stanford University, California.

Bassiliades, N., Antoniou, G., & Vlahavas, I. (2006). A Defeasible Logic Reasoner for the Semantic Web. *Int. Journal on Semantic Web and Information Systems*, 2, 1, 1-41.

Bassiliades N., & Vlahavas I. (2006). R-DEVICE: An Object-Oriented Knowledge Base System for RDF Metadata. *Int. Journal on Semantic Web and Information Systems, 2*, 2, 24-90.

Berners-Lee, T., Hendler J., & Lassila O. (2001). The Semantic Web. *Scientific American, 284*, 5, 34-43.

Bieber, P., & Cuppens, F. (1993). Expression of Confidentiality Policies with Deontic Logic. In J. C. Meyer & R. J. Wieringa (Eds.), *Deontic Logic in Computer Science: Normative System Specification* (pp. 103-123), Chichester, UK: John Wiley and Sons Ltd.

Blackburn, P., de Rijke, M., & Venema, Y. (2001). *Modal Logic*. Cambridge University Press, ISBN: 0521527147.

Boley, H. (2004). An Integrated Positional-Slotted Language for Semantic Web Knowledge. http://www.ruleml.org/submission/ruleml-shortation.html, last access: November 2010.

Boley, H. (2006). The RuleML Family of Web Rule Languages: Invited talk. *Proceedings of Fourth Workshop on Principles and Practice of Semantic Web Reasoning*, Springer LNCS 4187, pp.1-15, Budva.

Bonatti, P.A., Duma, C., Fuchs, N., Nejdl, W., Olmedilla, D., Peer, J., & Shahmehri, N. (2006). Semantic Web Policies - a Discussion of Requirements and Research Issues. *Proceedings of the 3rd European Semantic Web Conference*, LNCS vol. 4011, Budva, Montenegro, Springer.

Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., & van der Torre, L. (2001). The BOID Architecture: Conflicts between Beliefs, Obligations, Intentions and Desires. *Proceedings of the 5th Int. Conference on Autonomous Agents*, pp. 9-16, Montreal, Quebec, Canada, ACM.

Cohen, P. R., & Levesque, H. J. (1990). Intention is Choice with Commitment. *Artificial Intelligence, 42*, 2-3, 213-261.

Gottlob, G. (1992). Complexity Results for Nonmonotonic Logics. *Journal of Logic and Computation, 2*, 397-425.

Governatori, G. (2005). Representing Business Contracts in RuleML. *Int. Journal of Cooperative Information Systems, 14*, 2-3, 181-216.

Governatori, G., Hulstijn, J., Riveret, Ré. & Rotolo, A. (2007). Characterising Deadlines in Temporal Modal Defeasible Logic. *Proceedings of the 20th Australian Joint Conference on Artificial Intelligence (AI 2007)*, pp. 486-496.

Governatori, G., Padmanabhan, V., Rotolo, A. & Sattar, A. (2009). A Defeasible Logic for Modelling Policy-based Intentions and Motivational Attitudes. *Logic Journal of the IGPL, 17*, 3, 227-265.

Governatori, G., & Rotolo, A. (2004). Defeasible Logic: Agency, Intention and Obligation. *Proceedings of the 7th Int. Workshop on Deontic Logic in Computer Science*, pp. 114-128, Madeira, Portugal.

Governatori, G., & Rotolo, A. (2008a). BIO Logical Agents: Norms, Beliefs, Intentions in Defeasible Logic. *Journal of Autonomous Agents and Multi Agent Systems, 17*, 1, 36-69.

Governatori, G., & Rotolo, A. (2008b). A Computational Framework for Institutional Agency. *Artificial Intelligence and Law, 16*, 1, 25-52.

Hilpinen, R. (2001). Deontic Logic. In Goble & Lou (Eds.), *The Blackwell Guide to Philosophical Logic*, Blackwell.

Hofstadter, D. R. (1983). Metamagical Themas: Computer Tournaments of the Prisoner's Dilemma Suggest How Cooperation Evolves. *Scientific American, 248*, 5, 16-26.

Kagal, L., Berners-Lee, T., Connolly, D, & Weitzner, D. (2006). Using Semantic Web Technologies for Open Policy Management on the Web. *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006)*, July 16-20.

Kautz, H. A., & Selman, B. (1991). Hard Problems for Simple Default Theories. *Artificial Intelligence, 28*, 243-279.

Kolaczek, G. (2002). Application of Deontic Logic in Role-based Access Control. *Int. Journal of Applied Mathematics and Computer Science, 12*, 2, 269-275.

Kontopoulos, E., Bassiliades, N., Governatori, G., & Antoniou, G. (2008). Extending a Defeasible Reasoner with Modal and Deontic Logic Operators. *Proceedings of Workshop on Logics for Intelligent Agents and Multi-Agent Systems*

(WLIAMAS 2008) at IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 626-629, IEEE Computer Society Press, Sydney, Australia.

Kontopoulos, E., Bassiliades, N., Governatori, G., & Antoniou, G. (2011). Visualizing Semantic Web Proofs of Defeasible Logic in the DR-DEVICE System. *Knowledge-Based Systems*, 24(3), pp. 406-419.

Kravari, K., Kontopoulos, E., & Bassiliades, N. (2010a). EMERALD: A Multi-Agent System for Knowledge-based Reasoning Interoperability in the Semantic Web. *Proceedings of the 6th Hellenic Conference on Artificial Intelligence (SETN 2010)*, accepted for publication, Athens, Greece, 4-7 May.

Kravari, K., Kontopoulos, E., & Bassiliades, N. (2010b). Trusted Reasoning Services for Semantic Web Agents. *Informatica: International Journal of Computing and Informatics*, Slovenian Society Informatika, 34 (4), pp. 429-440.

Lam, H., & Governatori, G. (2009). The Making of SPINdle. In: Governatori, G., Hall, J., & Paschke, A. (Eds.) *2009 Int. Symposium on Rule Interchange and Applications*, LNCS, vol. 5858, pp. 315-322, Springer-Verlag, Berlin, Heidelberg.

Lokhorst, Gert -Jan C. (1996). Reasoning about actions and obligations in first-order logic. *Studia Logica*, 57(1), pp. 221-237.

Maher, M. J., & Governatori, G. (1999). A Semantic Decomposition of Defeasible Logics. *Proceedings of the 16th Nat. Conference on Artificial intelligence*, pp. 299-305, Orlando, American Assoc. for AI.

Meyer, J-J C. (2001). Epistemic Logic. In Goble & Lou (Eds.), *The Blackwell Guide to Philosophical Logic*, Blackwell.

Meyer, J-J C. (2003). Modal Epistemic and Doxastic Logic. In D. Gabbay & F. Guenthner (Eds.), *Handbook of Philosophical Logic (2nd edition)*, Vol. 10, Dordrecht: Kluwer, pp. 1-38.

Nute, D. (1994). Defeasible logic. In D. Gabbay (Ed.), *Handbook of Logic and Artificial Intelligence*, *3*, 353-395, Oxford University Press.

Nute, D. (1998). Norms, Priorities, and Defeasible Logic. In McNamara, P. & H. Prakken (Eds.), *Norms, Logics and Information Systems*, pp. 201-218, IOS Press, Amsterdam.

Pham, D. H., Governatori, G., Raboczi, S., Newman, A., & Thakur, S. (2008). On Extending RuleML for Modal Defeasible Logic. In N. Bassiliades, G. Governatori, & A. Paschke (Eds.), *Proceedings of the Int. Symposium on Rule Representation, Interchange and Reasoning on the Web (RuleML 2008)*, LNCS vol. 5321, pp. 89-103, Orlando, Florida, Springer-Verlag, Berlin, Heidelberg.

Rissland, E.L., & Skalak, D.B. (1991). CABARET: Rule Interpretation in a Hybrid Architecture. *Int. Journal of Man-Machine Studies*, *34*, 6, 839-887.

Riveret, Ré., Rotolo, A. & Governatori, G. (2007). Interaction between Normative Systems and Cognitive Agents in Temporal Modal Defeasible Logic. In: Boella, G., Torre, L. v. d. & Verhagen, H. (Eds.), *Normative Multi-Agent Systems (NorMAS)*, Dagstuhl, Germany.

Schild, U. J., & Herzog, S. (1993). The Use of Meta-rules in Rule Based Legal Computer Systems. *Proceedings of the 4th Int. Conference on Artificial Intelligence and Law (ICAIL'93)*, pp. 100-109, ACM Press.

Skylogiannis T., Antoniou G., Bassiliades N., Governatori G., & Bikakis A. (2007). DR-NEGOTIATE – A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies. *Data & Knowledge Engineering*, *63*, 2, 362-380.

Soininen, T., & Niemela, I. (1998). Developing a Declarative Rule Language for Applications in Product Configuration. In: Gupta, G. (Ed.), *Proceedings of Int. Symposium on Practical Aspects of Declarative Languages (PADL)*, pp. 305-319, Springer-Verlag.

Venema, Y. (2001). Temporal Logic. In Goble & Lou (Eds.), The Blackwell Guide to Philosophical Logic, Blackwell.