

On Extending RuleML for Modal Defeasible Logic

Guido Governatori¹, Duy Hoang Pham^{1,2}, Simon Raboczi²
Andrew Newman², Subhasis Thakur²

¹ National ICT Australia, Queensland Research Laboratory, Brisbane, Australia

² School of Information Technology and Electrical Engineering
The University of Queensland, Brisbane, Australia

Abstract. In this paper we present a general methodology to extend Defeasible Logic with modal operators. We motivate the reasons for this type of extension and we argue that the extension will allow for a robust knowledge framework in different application areas. The paper presents an extension of RuleML to capture Modal Defeasible Logic.

1 Introduction

Relations among organizations are guided by sets of rules or policies. A policy can define the privacy requirements of an user, access permissions for a resource, rights of an individual and so on. Many languages have been proposed to write policies. A few examples of these languages are P3P, XACML, SAML. These languages are XML based and use different tags to represent different information to be used in the description of a policy. The growth of the number of these languages and important, and the similarity of concepts these are trying to capture has recently led the W3C to create a special interest group on policy language [22] with the aim of providing a unifying approach to the representation of policies on the web.

A policy can be understood as a set of rules, and the purpose of policy languages (and rule languages in general) is to provide a medium to allow different stakeholders to achieve interoperability by exchanging their (relevant) policies. While the ability to exchange rules is very important, the real key issue is the ability to use and reason with rules in the same way. It might be possible that for some reasons the parties involved in an exchange or rules do not want to adopt the reasoning mechanism of their counterparts. However, they have to realise and understand how the counterparts are going to use the rules, and to consider this in their decision processes.

Rules and proofs are now part of the grand design of the Semantic Web. It has been recognised that the logic part –mainly understood as the OWL family and (fragment) of first order logic– has to be supplemented by rules. Thus the first problem we have to face is to combine logics for reasoning with rules and logics for reasoning with ontologies [23,8,7,14]. The second problem is that while there is only one classical first-order logic but there are many logics for reasoning with rules, and often these logics reflect different and sometimes incompatible facets of reasoning with rules. In addition, we are going to add modal operators and as we will argue in Section 2 even for the same interpretation of a modal operator different logical properties have been proposed. Thus we believe that if one wants to be able to share rules with others, it is of paramount importance

to be able to specify how to give meaning to the rules and the (modal) operators used in the rules, to enable users to process the information present in the rules in the same way.

The contribution of the paper is manifold. First we will argue that extending rule languages with modal operators offers a very powerful and rich environment to deal with situations where multiple parties are involved and intensional notions are required (Section 2). Deploying any reasoning mechanism for the Web faces an additional challenge: it has to have good computational properties. We defend and motivate our choices against this requirement in Sections 3 and 4. In Section 6.2 we will argue that a rule language should describe the elements of the language but in situations where there are many logics sharing the language, the rule language should provide facilities to describe the logic to be used to process the rules. Here we show how to extend RuleML to capture the descriptions characterising the extensions with modal operators indentified in Sections 5.1 and 5.2. In Section 7 we outline the implementation of the framework.

2 Modal Logics vs Modalities

Modal logic has been heavily used as a conceptual tool for establishing the foundations of the analysis of epistemic and doxastic notions (i.e., knowledge and belief) in terms of modal operators, paving thus the way to the field of agents and multi-agent systems. In this fields modal operators proved to be very powerful conceptual tools to describe the internal (mental) states of agents as well as interactions among agents. Deontic Logic is the modal logic where the modal operators are interpreted as is nowadays one of the most promising instruments for the formalisation of institutionalised organisation and the mutual relationships (normative position) among the actors in such models. Deontic Logic plays an important role in the formalisation of contracts [18,9].

What we want to stress out here is that modal logic is appropriate to provide a conceptual model for describing agents as well as many other intensional notions, in particular normative notions such as obligations, permissions, rights and so on which are important for policies, e-commerce and e-contract. Given this, the aim of this paper is to provide a computationally oriented non-monotonic rule based account of modal logic for the use and exchange of rules on the Web.

A modal operator qualifies the truth of the expressions it operates on, and many interpretations are possible for modal operator. Given the multiplicity of interpretations and the many facets of modalities, it is not possible to have a one size fits all (or most) situation. In general, there is no single modal logic even for a particular interpretation, and thus the designer of a particular application has to choose case by case which proprieties/principles are satisfied by the modal operators. The designer has to identify which notions are better modelled by modal operators and which are suitable to be captured by predicates.

Given the issues above, a supporter of modalities (particular *ad hoc* predicates whose interpretation is that of modal operators) might argue that modalities offer a more convenient approach since there is no need to create a new logic every time we have a new notion. Everything can be represented in first-order logic. After all, it is hard to distinguish between notions to be modelled by ordinary predicates and notions

to be modelled by modal operators. In addition, from a computational point of view first-order logic is semi-decidable while often modal logics are decidable, and there are examples where properties can be encoded easily in modal logic but they require high-order logic representations.

A first answer to this objection is that rather than adding ad hoc predicates to the language, improvements must be made by adding modal operators so as to achieve a richer language that can represent the behaviour of modal notions in a more natural and applicable manner. The advantage of this approach is to incorporate general and flexible reasoning mechanisms within the inferential engine.

A formal representation language should offer concepts close to the notions the language is designed to capture. For example, contracts typically contain provisions about deontic concepts such as obligations, permissions, entitlements, violations and other (mutual) normative positions that the signatories of a contract agree to comply with. Accordingly, a contract language should cater for those notions. In addition, the language should be supplemented by either a formal semantics or facilities to reason with and about the symbols of the language to give meaning to them. As usual, the symbols of the language can be partitioned in two classes: logical symbols and extra logical symbols. The logical symbols are meant to represent general concepts and structures common to every contract while extra logical symbols encode the specific subject matter of given contracts. In this perspective the notions of obligation and permission will be represented by deontic modalities while concepts such as price, service and so on are better captured by predicates since their meaning varies from contract to contract.

In general, we believe that the approach with modal operators is superior to the use of ad hoc predicates at least for the following aspects¹:

- *Ease of expression and comprehension.* In the modal approach the relationships among modal notions are encoded in the logic and reasoning mechanism while for ad hoc predicates knowledge bases are cluttered with rules describing the logical relationships among different modes/representations of one and the same concept. For example, in a set of rules meant to describe a contract, given the predicate $pay(X)$, we have to create predicates such as $obligatory_pay(X)$, $permitted_pay(X)$, ... and rules such as $obligatory_pay(X) \rightarrow permitted_pay(X)$ and so on. Thus ad hoc predicates do not allow users to focus only and exclusively on aspects related to the content of a contract, without having to deal with any aspects related to its implementation.
- *Clear and intuitive semantics.* It is possible to give a precise, unambiguous, intuitive and general semantics to the notions involved while each ad hoc predicate requires its own individual interpretation, and in some cases complex constructions (for example reification) are needed to interpret some ad hoc predicates.
- *Modularity.* A current line of research proposes that the combination of deontic operators with operators for speech acts and actions faithfully represent complex normative positions such as delegation, empowerment as well as many others that

¹ In addition to the aspects we discuss here, we would like to point out that it has been argued [13,15] that deontic logic is better than a predicate based representation of obligations and permissions when the possibility of norm violation is kept open. A logic of violation is essential for the representation of contracts where rules about violations are frequent [9].

may appear in contracts [16]. In the modal approach those aspects can be added or decomposed modularly without forcing the user to rewrite the predicates and rules to accommodate the new facilities, or to reason at different granularity.

3 Defeasible Logic

Defeasible Logic (DL) [20,1] is a simple, efficient but flexible non-monotonic formalism that can deal with many different intuitions of non-monotonic reasoning [2], and efficient and powerful implementations have been proposed [19,4]. In the last few years the logic and its variants have been applied in many fields.

Knowledge in DL can be represented in two ways: facts and rules.

Facts are indisputable statements, represented either in form of states of affairs (literal and modal literal) and actions that have been performed. Facts are represented by predicates. For example, “the price of the spam filter is \$50” is represented by

$$Price(SpamFilter, 50).$$

A *rule*, on the other hand, describes the relationship between a set of literals (premises) and a literal (conclusion), and we can specify how strong the relationship is and the mode the rule connects the antecedent and the conclusion. As usual, rules allow us to derive new conclusions given a set of premises. Since rules have a mode, the conclusions will be modal literals. As far as the strength of rules is concerned we distinguish between *strict rules*, *defeasible rules* and *defeaters*; for the mode we have one set of rules (base rules) describing the inference principles of the basic logic plus one mode for each modal operator of the language (modal rules). As we will see, the idea of modal rules is to introduce modalised conclusions. Accordingly, if we have a modal rule for p for a modal operator \Box_i , this means that the rule allows for the derivation of $\Box_i p$.

Strict rules, defeasible rules and defeaters are represented, respectively, by expressions of the form $A_1, \dots, A_n \rightarrow B$, $A_1, \dots, A_n \Rightarrow B$ and $A_1, \dots, A_n \rightsquigarrow B$, where A_1, \dots, A_n is a possibly empty set of prerequisites and B is the conclusion of the rule. We only consider rules that are essentially propositional. Rules containing free variables are interpreted as the set of their ground instances.

Strict rules are rules in the classical sense: whenever the premises are indisputable then so is the conclusion. Thus, they can be used for definitional clauses. An example of a strict rule is “A ‘Premium Customer’ is a customer who has spent \$10000 on goods”:

$$TotalExpense(X, 10000) \rightarrow PremiumCustomer(X).$$

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is “Premium Customer are entitled to a 5% discount”:

$$PremiumCustomer(X) \Rightarrow Discount(X).$$

The idea is that if we know that someone is a Premium Customer then we may conclude that she is entitled to a discount *unless there is other evidence suggesting that she may not be* (for example if she buys a good in promotion).

Defeaters are a special kind of rules. They are used to prevent conclusions not to support them. For example:

$$SpecialOrder(X), PremiumCustomer(X) \rightsquigarrow \neg Surcharge(X).$$

This rule states that premium customers placing special orders might be exempt from the special order surcharge. This rule can prevent the derivation of a “surcharge” conclusion. However, it cannot be used to support a “not surcharge” conclusion.

DL is a “skeptical” non-monotonic logic, meaning that it does not support contradictory conclusions.² Instead, DL seeks to resolve conflicts. In cases where there is some support for concluding A but also support for concluding $\neg A$, DL does not conclude neither of them (thus the name “skeptical”). If the support for A has priority over the support for $\neg A$ then A is concluded.

As we have alluded to above, no conclusion can be drawn from conflicting rules in DL unless these rules are prioritised. The *superiority relation* is used to define priorities among rules, that is, where one rule may override the conclusion of another rule. For example, given the defeasible rules

$$\begin{aligned} r &: PremiumCustomer(X) \Rightarrow Discount(X) \\ r' &: SpecialOrder(X) \Rightarrow \neg Discount(X) \end{aligned}$$

which contradict one another, no conclusive decision can be made about whether a Premium Customer, who has placed a special order, is entitled to the 5% discount. But if we introduce a superiority relation $>$ with $r' > r$, we can indeed conclude that special orders are not subject to discount.

We now give a short informal presentation of how conclusions are drawn in DL. Let D be a theory in DL (i.e., a collection of facts, rules and a superiority relation). A *conclusion* of D is a tagged literal and can have one of the following four forms:

- + Δq meaning that q is definitely provable in D (i.e., using only facts and strict rules).
- Δq meaning that we have proved that q is not definitely provable in D .
- + ∂q meaning that q is defeasibly provable in D .
- ∂q meaning that we have proved that q is not defeasibly provable in D .

Strict derivations are obtained by forward chaining of strict rules while a defeasible conclusion p can be derived if there is a rule whose conclusion is p , whose prerequisites (antecedent) have either already been proved or given in the case at hand (i.e. facts), and any stronger rule whose conclusion is $\neg p$ has prerequisites that fail to be derived. In other words, a conclusion p is derivable when:

- p is a fact; or
- there is an applicable strict or defeasible rule for p , and either
 - all the rules for $\neg p$ are discarded (i.e., are proved to be not applicable) or
 - every applicable rule for $\neg p$ is weaker than an applicable strict³ or defeasible rule for p .

The formal definitions of derivations in DL are in the next section.

² To be precise contradictions can be obtained from the monotonic part of a defeasible theory, i.e., from facts and strict rules.

³ Notice that a strict rule can be defeated only when its antecedent is defeasibly provable.

4 Modal Defeasible Logic

As we have seen in Section 1, modal logics have been put forward to capture many different notions somehow related to the intensional nature of agency as well as many other notions. Usually modal logics are extensions of classical propositional logic with some intensional operators. Thus, any modal logic should account for two components: (1) the underlying logical structure of the propositional base and (2) the logic behaviour of the modal operators. Alas, as is well-known, classical propositional logic is not well suited to deal with real life scenarios. The main reason is that the descriptions of real-life cases are, very often, partial and somewhat unreliable. In such circumstances, classical propositional logic might produce counterintuitive results insofar as it requires complete, consistent and reliable information. Hence any modal logic based on classical propositional logic is doomed to suffer from the same problems.

On the other hand, the logic should specify how modalities can be introduced and manipulated. Some common rules for modalities are, e.g., Necessitation (from $\vdash \phi$ infer $\vdash \Box \phi$) and RM (from $\vdash \phi \rightarrow \psi$ infer $\vdash \Box \phi \rightarrow \Box \psi$). Both dictates conditions to introduce modalities purely based on the derivability and structure of the antecedent. These rules are related to the well-known problem of logical omniscience and put unrealistic assumptions on the capability of an agent. However, if we take a constructive interpretation, we have that if an agent can build a derivation of ϕ then she can build a derivation of $\Box \phi$. We want to maintain this intuition here, but we want to replace derivability in classical logic with a practical and feasible notion like derivability in DL. Thus, the intuition behind this work is that we are allowed to derive $\Box_i p$ if we can prove p with the mode \Box_i in DL.

To extend DL with modal operators we have two options: 1) to use the same inferential mechanism as basic DL and to represent explicitly the modal operators in the conclusion of rules [21]; 2) introduce new types of rules for the modal operators to differentiate between modal and factual rules.

For example, the “deontic” statement “The Purchaser shall follow the Supplier price lists” can be represented as

$$\text{AdvertisedPrice}(X) \Rightarrow O_{\text{purchaser}} \text{Pay}(X)$$

if we follow the first option and

$$\text{AdvertisedPrice}(X) \Rightarrow_{O_{\text{purchaser}}} \text{Pay}(X)$$

according to the second option, where $\Rightarrow_{O_{\text{purchaser}}}$ denotes a new type of defeasible rule relative to the modal operator $O_{\text{purchaser}}$. Here, $O_{\text{purchaser}}$ is the deontic “obligation” operator parametrised to an actor/role/agent, in this case the purchaser.

The differences between the two approaches, besides the fact that in the first approach there is only one type of rules while the second accounts for factual and modal rules, is that the first approach has to introduce the definition of p -incompatible literals (i.e., a set of literals that cannot be hold when p holds) for every literal p . For example, we can have a modal logic where $\Box p$ and $\neg p$ cannot be both true at the same time. Moreover, the first approach is less flexible than the second: in particular in some cases it must account for rules to derive $\Diamond p$ from $\Box p$; similarly conversions (see Section 5.2)

require additional operational rules in a theory, thus the second approach seems to offer a more conceptual tool than the first one. The second approach can use different proof conditions based on the modal rules to offer a more fine grained control over the modal operators and it allows for interaction between modal operators.

As usual with non-monotonic reasoning, we have to specify 1) how to represent a knowledge base and 2) the inference mechanism used to reason with the knowledge base. The language of Modal Defeasible Logic consists of a finite set of modal operators $Mod = \{\Box_1, \dots, \Box_n\}$ and a (numerable) set of atomic propositions $Prop = \{p, q, \dots\}$.⁴

We supplement the usual definition of literal (an atomic proposition or the negation of it), with the following clauses

- if l is a literal then $\Box_i l$, and $\neg \Box_i l$, are literals if l is different from $\Box_i m$, and $\neg \Box_i m$, for some literal m .

The above condition prevents us from having sequences of modalities where we have successive occurrences of one and the same modality; however, iterations like $\Box_i \Box_j$ and $\Box_i \Box_j \Box_i$ are legal in the language.

Given a literal l with $\sim l$ we denote the complement of l , that is, if l is a positive literal p then $\sim l = \neg p$, and if $l = \neg p$ then $\sim l = p$.

According to the previous discussion a Modal Defeasible Theory D is a structure (F, R, \succ) where F is a set of facts (literals or modal literals), $R = R^B \cup \bigcup_{1 \leq i \leq n} R^{\Box_i}$, where R^B is the set of base (un-modalised) rules, and each R^{\Box_i} is the set of rules for \Box_i and $\succ \subseteq R \times R$ is the superiority relation. A rule r is an expression $A(r) \hookrightarrow_X C(r)$ such that ($\hookrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$, X is B , for a base rule, and a modal operator otherwise), $A(r)$ the antecedent or body of r is a (possible empty) set of literals and modal literals, and $C(r)$, the consequent or head of r is a literal if r is a base rule and either a literal or a modal literal Yl where Y is a modal operator different from X . Given a set of rules R we use R_{sd} to denote the set of strict and defeasible rules in R , and $R[q]$ for the set of rules in R whose head is q .

The derivation tags are now indexed with modal operators. Let X range over Mod . A conclusion can now have the following forms:

- $+\Delta_X q$: q is definitely provable with mode X in D (i.e., using only facts and strict rules of mode X).
- $-\Delta_X q$: we have proved that q is not definitely provable with mode X in D .
- $+\partial_X q$: q is defeasibly provable with mode X in D .
- $-\partial_X q$: we have proved that q is not defeasibly provable with mode X in D .

Then if we can prove $+\partial_{\Box_i} q$, then we can assert $\Box_i q$.

Formally provability is based on the concept of a *derivation* (or proof) in D . A derivation is a finite sequence $P = (P(1), \dots, P(n))$ of tagged literals satisfying the proof conditions (which correspond to inference rules for each of the kinds of conclusion). $P(1..n)$ denotes the initial part of the sequence P of length n .

⁴ The language can be extended to deal with other notions. For example to model agents, we have to include a (finite) set of agents, and then the modal operators can be parameterised with the agents. For a logic of action or planning, it might be appropriate to add a set of atomic actions/plans, and so on depending on the intended applications.

Before introducing the proof conditions for the proof tags relevant to this paper we provide some auxiliary notions.

Let $\#$ be either Δ or ∂ . Given a proof $P = (P(1), \dots, P(n))$ in D and a literal q we will say that q is Δ -provable in P , or simply Δ -provable, if there is a line $P(m)$ of the derivation such that either:

1. if $q = l$ then
 - $P(m) = +\#l$ or
 - $\Box_i l$ is $\#$ -provable in $P(1..m-1)$ and \Box_i is reflexive⁵
2. if $q = \Box_i l$ then
 - $P(m) = +\#_i l$ or
 - $\Box_j \Box_i l$ is $\#$ -provable in $P(1..m-1)$, for some $j \neq i$ such that \Box_j is reflexive.
3. if $q = \neg \Box_i l$ then
 - $P(m) = -\#_i l$ or
 - $\Box_j \neg \Box_i l$ is $\#$ -provable in $P(1..m-1)$, for some $j \neq i$ such that \Box_j is reflexive.

In a similar way we can define a literal to be Δ - and ∂ -rejected by taking, respectively, the definition of Δ -provable and ∂ -provable and changing all positive proof tags into negative proof tags, adding a negation in front of the literal when the literal is prefixed by a modal operator \Box_j , and replacing all the *ors* by *ands*. Thus, for example, we can say that a literal $\Box_i l$ is ∂ -rejected if, in a derivation, we have a line $-\partial_i l$, and the literal $\neg \Box_i \neg l$ is ∂ -rejected if we have $+\partial_i \neg l$ and so on.

Let X be a modal operator and $\#$ is either Δ or ∂ . A literal l is $\#_X$ -provable if the modal literal Xl is $\#$ -provable; l is $\#_X$ -rejected if the literal Xl is $\#$ -rejected.

Based on the above definition of provable and rejected literals we can give the conditions to determine whether a rule is applicable or the rule cannot be used to derive a conclusion (i.e., the rule is discarded).

The proof conditions for $+\Delta$ correspond to monotonic forward chaining of derivations and, for space limitations are not given here (see [1,10] for the definitions).

Let X be a modal operator or B . Given a rule r we will say that the rule is ∂_X -applicable iff

1. $r \in R^X$ and $\forall a_k \in A(r)$, a_k is ∂ -provable; or
2. if $X \neq B$ and $r \in R^B$, i.e., r is a base rule, then $\forall a_k$, a_k is ∂_X -provable.

Given a rule r we will say that the rule is ∂_X -discarded iff

1. $r \in R^X$ and $\exists a_k \in A(r)$, a_k is ∂ -rejected; or
2. if $X \neq B$ and $r \in R^B$, i.e., r is a base rule, then $\exists a_k$, a_k is ∂_X -rejected.

We give now the proof condition for defeasible conclusions (i.e., conclusions whose tag is $+\partial$). Defeasible derivations have an argumentation like structure divided in three phases. In the first phase, we put forward a supported reason (rule) for the conclusion we want to prove. Then in the second phase, we consider all possible (actual and not) reasons against the desired conclusion. Finally, in the last phase, we have to rebut all

⁵ A modal operator \Box_i is reflexive iff the truth of $\Box_i \phi$ implies the truth of ϕ . In other words \Box_i is reflexive when we have the modal axiom $\Box_i \phi \rightarrow \phi$.

the counterarguments. This can be done in two ways: we can show that some of the premises of a counterargument do not obtain, or we can show that the argument is weaker than an argument in favour of the conclusion. This is formalised by the following (constructive) proof conditions.

- $+ \partial_X$: If $P(n+1) = + \partial_X q$ then
- 1) $+ \Delta_X q \in P(1..n)$, or
 - 2) $- \Delta_X \sim q \in P(1..n)$ and
 - 2.1) $\exists r \in R_{sd}[q]$: r is ∂_X -applicable and
 - 2.2) $\forall s \in R[\sim q]$ either s is ∂_X -discarded or
 $\exists w \in R[q]$: w is ∂_X -applicable and $w \succ s$.

The above condition is, essentially, the usual condition for defeasible derivations in DL, we refer the reader to [20,1,10] for more thorough treatments. The only point we want to highlight here is that base rules can play the role of modal rules when all the literals in the body are ∂_{\square_i} -derivable. Thus, from a base rule $a, b \Rightarrow_B c$ we can derive $+ \partial_{\square_i} c$ if both $+ \partial_{\square_i} a$ and $+ \partial_{\square_i} b$ are derivable while this is not possible using the rule $a, \square_i b \Rightarrow_B c$ (see Section 5.2).

5 Modal Defeasible Logic with Interactions

Notice that the proof condition for $+ \partial$ given in Section 3 and then those for the other proof tags are the same as those of basic DL as given in [1]. What we have done is essentially to consider $n+1$ non-monotonic consequence relation defined in DL and compute them in parallel. In the previous sections, we have argued that one of the advantages of modal logic is the ability to deal with complex notions composed by several modalities, or by interactions of modal operators. Thus, we have to provide facilities to represent such interactions. In Modal DL it is possible to distinguish two types of interactions: conflicts and conversions. In the next two sections, we will motivate them and we show how to capture them in our framework.

5.1 Conflicts

Let us take a simple inclusion axiom of multi-modal logic relating two modal operators \square_1 and \square_2 : $\square_1 \phi \rightarrow \square_2 \phi$. The meaning of this axiom is that every time we are able to prove $\square_1 \phi$, then we are able to prove $\square_2 \phi$. Thus, given the intended reading of the modal operators in our approach –a modal operator characterises a derivation using a particular mode, it enables us to transform a derivation of $\square_1 \phi$ into a derivation of $\square_2 \phi$. If the logic is consistent, we also have that $\square_1 \phi \rightarrow \square_2 \phi$ implies that it is not possible to prove $\square_2 \neg \phi$ given $\square_1 \phi$, i.e., $\square_1 \phi \rightarrow \neg \square_2 \neg \phi$. However, this idea is better illustrated by the classically equivalent formula $\square_1 \phi \wedge \square_2 \neg \phi \rightarrow \perp$. When the latter is expressed in form of the inference rule

$$\frac{\square_1 \phi, \square_2 \neg \phi}{\perp} \quad (1)$$

it suggests that it is not possible to obtain $\square_1 \phi$ and $\square_2 \neg \phi$ together. This does not mean that $\square_1 \phi$ implies $\square_2 \phi$, but that the modal operators \square_1 and \square_2 are in conflict with each

other. Modal DL is able to differentiate between the two formulations: For the inclusion version (i.e., $\Box_1\phi \rightarrow \Box_2\phi$) what we have to do is just to add the following clause to the proof conditions for $+\partial_{\Box_2}$ (and the other proof tags accordingly) with the condition

$$+\partial_{\Box_1}q \in P(1..n)$$

For the second case (i.e., $\Box_1\phi \wedge \neg\Box_2\phi \rightarrow \perp$), we have to give a preliminary definition.

Given a modal operator \Box_i , $\mathcal{F}(\Box_i)$ is the set of modal operators in conflict with \Box_i . If the only conflict axiom we have is $\Box_1\phi \wedge \Box_2\phi \rightarrow \perp$ then $\mathcal{F}(\Box_1) = \{\Box_2\}$. With $R^{\mathcal{F}(\Box_i)}$ we denote the union of rules in all R^{\Box_j} where $\Box_j \in \mathcal{F}(\Box_i)$. At this point to implement the proof condition for the conflict all we have to do is to replace clause 2.2 of the definition of $+\partial_{\Box_i}q$ with the clause

$$2.2)\forall s \in R^{\mathcal{F}(\Box_i)}[\sim q] \text{ either } s \text{ is } \partial_X\text{-discarded or} \\ \exists w \in R[q]: w \text{ is } \partial_X\text{-applicable and } w \succ s.$$

The notion of conflict has been proved useful in the area of cognitive agents, i.e., agent whose rational behaviour is described in terms of mental and motivational attitudes including beliefs, intentions, desires and obligations. Classically, agent types are characterised by stating conflict resolution methods in terms of orders of overruling between rules [6,10]. For example, an agent is *realistic* when rules for beliefs override all other components; she is *social* when obligations are stronger than the other components with the exception of beliefs. Agent types can be characterised by stating that, for any types of rules X and Y , for every r and r' , $r \in R^X[q]$ and $r' \in R^Y[\sim q]$, we have that $r > r'$.

5.2 Conversions

Another interesting feature that could be explained using our formalism is that of *rule conversion*. Indeed, this feature allows us to model the interactions between different modal operators. In general, notice that in many formalisms it is possible to convert from one type of conclusion into a different one. For example, the right weakening rule of non-monotonic consequence relations (see [17])

$$\frac{B \vdash C \quad A \vdash B}{A \vdash C}$$

allows the combination of non-monotonic and classical consequences.

Suppose that a rule of a specific type is given and all the literals in the antecedent of the rule are provable in one and the same modality. If so, is it possible to argue that the conclusion of the rule inherits the modality of the antecedent? To give an example, suppose we have that $p, q \Rightarrow_{\Box_i} r$ and that we obtain $+\partial_{\Box_j}p$ and $+\partial_{\Box_j}q$. Can we conclude $\Box_j r$? In many cases this is a reasonable conclusion to obtain.

For this feature we have to declare which modal operators can be converted and the target of the conversion. Given a modal operator \Box_i , with $\mathcal{V}(\Box_i)$ we denote the set of modal operators \Box_j that can be converted to \Box_i . In addition, we assume that base rules can be converted to all other types of rules. The condition to have a successful conversion of a rule for \Box_j into a rule for \Box_i is that all literals in the antecedent of the rules are provable modalised with \Box_i . Formally we have thus to add (disjunctively) in the support phase (clause 2.1) of the proof condition for ∂_{\Box_i} the following clause

2.1b) $\exists r \in R^{\mathcal{Y}(\Box_i)}[q]$ such that r is ∂_{\Box_i} -applicable

The notion of conversion enables us to define new interesting agent types [10].

We conclude this section with a formalisation of the Yale Shooting Problem that illustrates the notion of conversion. Let INT be the modal operator for intention. The Yale Shooting Problem can be described as follows⁶

$$liveAmmo, load, shoot \Rightarrow_B kill$$

This rule encodes the knowledge of an agent that knows that loading the gun with live ammunitions, and then shooting will kill her friend. This example clearly shows that the qualification of the conclusions depends on the modalities relative to the individual acts “load” and “shoot”. In particular, if the agent intends to load and shoot the gun (INT(*load*), INT(*shoot*)), then, since she knows that the consequence of these actions is the death of her friend, she intends to kill him ($+\partial_{INT}kill$). However, in the case she has the intention to load the gun ($+\partial_{INT}load$) and for some reason shoot it (*shoot*), then the friend is still alive ($-\partial kill$).

6 RuleML

Starting with the RuleML 0.91 XML Schema for Datalog with classical negation, we extended the syntax to support defeasible rules and modal operators.

6.1 Defeasible Rule Markup

RuleML already supports strict rules via the `Implies` element and allows them to be named using the `oid` element. We need to extend the syntax to express defeasible rules, defeaters, and superiority relations.

To add defeasible rules and defeaters as described in §3, we borrow syntax from the DR-DEVICE rule language [4]. We add a `@ruletype` attribute to the `Implies` element, allowing it to take one of three values: `strictrule`, `defeasiblerule` or `defeater`. Because `strictrule` is implied when `@ruletype` is absent, when non-defeasible RuleML rulesets are imported their rules are correctly considered strict.

DR-DEVICE expresses the superiority relation by using the `@superior` attribute on the superior rule as a link to the `@ruleID` label of the inferior rule. We found this unsuitable because we may need to mark a rule as superior to more than one other rule, and an XML element can only bear a single `@superior` attribute. Using the scheme from [9, §5] instead, we explicitly represent the superiority relation using the distinguished predicate `Override`.

6.2 Modal Operator Markup

In §2 we argued against modality predicates such as those proposed in [5, §4]. Furthermore, in §4 we proposed two alternatives, modal operators and modal rules.

⁶ Here we will ignore all temporal aspects and we will assume that the sequence of actions is done in the correct order.

To support the first alternative, we introduce a Mode element. The @modetype attribute is a URI-valued identifier for the intended semantics of the modal operator, e.g. necessity, belief, obligation. The Mode may optionally contain a single parameters element whose zero or more children are used to further distinguish modes, e.g. between the beliefs of various agents, or between time instants in the case of a temporal operator. Two modes are identical if their @modetype and all their parameters are equal. For example, $r1 : AdvertisisedPrice(X) \Rightarrow O_{purchaser}Pay(X)$ is represented as

```
<Implies ruletype="defeasiblerule">
  <oid><Ind>r1</Ind></oid>
  <head>
    <Mode modetype="http://www.example.org/obligation">
      <parameters>
        <Ind>purchaser</Ind>
      </parameters>
      <Atom><Rel>Pay</Rel><Var>X</Var></Atom>
    </Mode>
  </head>
  <body>
    <Atom><Rel>AdvertisisedPrice</Rel><Var>X</Var></Atom>
  </body>
</Implies>
```

To support the second alternative, modal rules, we introduce a mode element which may appear as a child of the Implies element. It requires the same @modetype attribute as the Mode element. Its zero or more children distinguish the mode in the same way as the children of a Mode's parameters. For example, $r2 : AdvertisisedPrice(X) \Rightarrow O_{purchaser}Pay(X)$ is represented as

```
<Implies ruletype="defeasiblerule">
  <oid><Ind>r2</Ind></oid>
  <mode modetype="http://www.example.org/obligation">
    <Ind>purchaser</Ind>
  </mode>
  <head>
    <Atom><Rel>Pay</Rel><Var>X</Var></Atom>
  </head>
  <body>
    <Atom><Rel>AdvertisisedPrice</Rel><Var>X</Var></Atom>
  </body>
</Implies>
```

6.3 Modal Interactions

The conflict and conversion interactions introduced in §5 are not represented in RuleML. Instead, we express them in a separate document with its own custom XML Schema. This is an additional input file used to configure the reasoner. It lists the supported modes, identifying them globally using the same URIs referenced by the @modetype attributes in the rules, and locally to the document with short XML IDs. These IDs are then used to succinctly list any conflict sets and conversion pairs. The following shows

an example configuration file for a ‘social’ agent [10], that is an agent whose obligations prevail over her intentions and beliefs can be used to derive non primitive intentions and obligations.

```
<?xml version="1.0" encoding="UTF-8"?>
<ModeSet xmlns="http://www.example.org/modeset-ns"
  xmlns:ruleml="http://www.ruleml.org/0.91/xsd"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.org/xsd/ruleset.xsd">
  <Mode id="BEL1" href="http://www.example.org/mode/belief">
    <ruleml:Ind>agent1</ruleml:Ind>
  </Mode>
  <Mode id="OBL" href="http://www.example.org/mode/obligation"/>
  <Mode id="INT1" href="http://www.example.org/mode/intention">
    <ruleml:Ind>agent1</ruleml:Ind>
  </Mode>
  <Conflict between="OBL INT1"/>
  <Conversion from="BEL1" to="INT1"/>
  <Conversion from="BEL1" to="OBL"/>
</ModeSet>
```

7 Implementation

The reasoning process of Modal DL has three phases. In the pre-processing phase, the theory in the RuleML format are loaded into the mechanism and is transformed into an equivalent theory without superiority relation and defeaters. In the next phase, the rule loader, which parses the theory obtained in the first phase, generates the data structure for the inferential phase. Finally, the inference engine applies modifications to the data structure, where at every step it reduces the complexity of the data structure.

Theory transformation: The transformation operates in three steps. The first two steps remove the defeaters rules and the superiority relation among rules by applying the transformations similar to those of [10]. Essentially, the hierarchy of the modal operators is generated from the conflicting relationship among these operators. The modal operator on the top of the hierarchy plays the role of the *BEL* operator as in [10]. This amounts to take the rules for the modal operator at the top of the hierarchy as the set of base rules. The third step performs conversions of every modal rule into a rule with a new modal operator as specified by the theory.

Rule loader: The rule loader creates a data structure as follows: for every (modal) literal in the theory, we create an entry whose structure includes:

- a list of (pointers to) rules having the literal in the head. In order to simplify the data structure, a modal literal from the head of a rule is built from the head atom and the modal operator of the corresponding rule.
- a list of (pointers to) rules having the literal in the body

- a list of (pointers to) entries of complements of the literal. Notice that the complements of a literal should take into account of the occurrence the modal operator. For example, the complements of the literal $\Box_i l$ are $\neg\Box_i l$ and $\Box_i \sim l$; if the operator is reflexive we have to include also l as a complement of $\Box_i l$.
- a list of entries of literals which conflict with the literal. The conflict relationship is derived from the conflicting modal operators dictated by the theory. In addition, a modal literal $\Box_i l$ always conflicts with $\sim l$ when \Box_i is reflexive.

In order to improve the computational performance, every list in the data structure is implemented as a hash table.

Inferential engine: The Engine is based on an extension of the Delores algorithm proposed in [19] as a computational model of Basic Defeasible Logic. In turn, the engine

- Assert each fact (as a literal) as a conclusion and removes the literal from the rules, where the literal positively occurs in the body, and “deactivate” the rules where either its complements or its conflicting literals occur in the body.
- Scan the list of active rules for rules with the empty body. Take the (modal) literal from the head, remove the rule, and put the literal into the pending facts. The literal is removed from the pending facts and adds to the list of facts if either there is no such rule (of the appropriate type) whose head contains the complements of the literal or literals with conflicting modes, or it is impossible to prove these literals.
- It repeats the first step.
- The algorithm terminates when one of the two steps fails.⁷ On termination, the algorithm outputs the set of conclusions from the list of facts in the RuleML format.

8 Conclusion

To sum up the contribution of the paper is manifold. We have argued that rule languages for the Semantic Web can benefit from modal extensions. However, given the multiplicity of interpretations of modal operators (as well as the different intuition behind execution model of rule systems) present a further challenge. An interchange language should be able to provide not only the syntax to represent rule, but it should provide facilities to describe how the rules should be processed (i.e., what the is the logic to be used to interpret the rules). On this respect we have identified the basic mechanisms to relate modal operators in a rule language (conflict and conversion).

The framework we have outlined in the previous sections has proven robust enough to represent and reason with different scenarios and applications, from business contracts [9] to normative reasoning [12], policy based cognitive agents [10] and workflow systems [11]. The main reason of the success, we believe, is due to the fact that Modal DL conceptually strengthen the expressive power of DL with modal operators, but at the same time it maintains the constructive and computational flavour of DL. Indeed, we have proved that the complexity of Modal DL as outlined here is linear [10]. This makes the logic very attractive from the knowledge representation point of view.

⁷ This algorithm outputs $+\partial$; $-\partial$ can be computed by an algorithm similar to this with the “dual actions”. For $+\Delta$ we have just to consider similar constructions where we examine only the first parts of step 1 and 2. $-\Delta$ follows from $+\Delta$ by taking the dual actions.

References

1. G. Antoniou, D. Billington, G. Governatori, and Michael J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, 2001.
2. G. Antoniou, D. Billington, G. Governatori, Michael J. Maher, and A. Rock. A family of defeasible reasoning logics and its implementation. In *Proc. ECAI 2000*: 459–463. IOS Press, 2000.
3. G. Antoniou and H. Boley, editors. *Proc. RuleML 2004*. LNCS 3323. Springer, 2004.
4. N. Bassiliades, G. Antoniou, and I.P. Vlahavas. A defeasible logic reasoner for the semantic web. *International Journal on Semantic Web and Information Systems*, 1(2):1–41, 2006.
5. H. Boley. The RuleML family of web rule languages. In *Proc 4th PPSWR*: 1–17, Springer 2006.
6. J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.
7. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-founded semantics for description logic programs in the semantic web. In [3]: 81–97.
8. G. Governatori. Defeasible description logics. In [3]: 98–112.
9. G. Governatori. Representing business contracts in RuleML. *International Journal of Co-operative Information Systems*, 14(2-3):181–216, 2005.
10. G. Governatori and A. Rotolo. BIO Logical Agents: Norms, Beliefs, Intentions in Defeasible Logic. *Journal of Autonomous Agents and Multi-Agents*, 2008.
11. G. Governatori, A. Rotolo, and S. Sadiq. A model of dynamic resource allocation in workflow systems. In *Database Technology 2004*, CRPIT 27: 197–206. ACS, 2004.
12. G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *Proc. ICAIL 2005*: 25–34. ACM Press, 2005.
13. H. Herrestad. Norms and formalization. In *Proc. ICAIL'91*: 175–184. ACM Press, 1991.
14. I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosz, and M. Dean. SWRL: A semantic web rule language combining owl and ruleml. W3C Member Submission, 21 May 2004. Available at <http://www.w3.org/Submission/SWRL/>.
15. A.J.I. Jones and M. Sergot. On the characterization of law and computer systems: the normative systems perspective. In *Deontic logic in computer science: normative system specification*: 275–307. John Wiley and Sons Ltd., 1993.
16. A.J.I. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):429–445, 1996.
17. S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.
18. R.M. Lee. A logic model for electronic contracting. *Decision Support Systems*, 4:27–44, 1988.
19. M.J. Maher, A. Rock, G. Antoniou, D. Billington, and T. Miller. Efficient defeasible reasoning systems. *International Journal of Artificial Intelligence Tools*, 10(4):483–501, 2001.
20. D. Nute. Defeasible logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 353–395. Oxford University Press, 1994.
21. D. Nute. Norms, priorities and defeasibility. In *Norms, Logics and Information Systems. New Studies in Deontic Logic*: 83–100. IOS Press, 1998.
22. Pling – w3c policy languages interest group. <http://www.w3.org/Policy/pling/>, 2007. Accessed, November 1, 2007.
23. K. Wang, D. Billington, J. Blee, and G. Antoniou. Combining description logic and defeasible logic for the semantic web. In [3]: 170–181.