A Modelling and Reasoning Framework for Social Networks Policies

Guido Governatori[1] and Renato Iannella[2]
[1] NICTA, Queensland Research Laboratory, Australia
guido.governatori@nicta.com.au
[2] Semantic Identity, Brisbane, Australia
ri@semanticidentity.com

Abstract

Policy languages (such as privacy and rights) have had little impact on the wider community. Now that Social Networks have taken off, the need to revisit Policy languages and realign them towards Social Networks requirements has become more apparent. One such language is explored as to its applicability to the Social Networks masses. We also argue that policy languages alone are not sufficient and thus they should be paired with reasoning mechanisms to provide precise and unambiguous execution models of the policies. To this end we propose a computationally oriented model to represent, reason with and execute policies for Social Networks.

Keywords Social Networks; Open Digital Rights Language (ODRL); Policy; Privacy; Rights; Defeasible Logic; Formal Contract Language (FCL)

1 Introduction

The Web undoubtedly has developed an impressive collection of technologies for supporting sophisticated information sharing and representation. Social Networks – via the innovative use of Web 2.0 features – have also taken the wider web community by surprise with such rapid uptake and widespread sharing of user-generated content.

A major lesson from Social Networks is that by offering "simplicity and efficiency" we can attract mass audiences (Mikroyannidis 2007). Equally, a major lesson from the Semantic Web is that we can "deliver information directly to people for whom the information was relevant" by adopting a semantically-aware social networking stack across Social Network services (Breslin and Decker 2007). The challenge now is to bring these two communities together in such a way that the technology (e.g., Semantic Web) meets the needs of a mass audience (e.g., Social Networks).

Social Networks have highlighted one particular area of concern:

> "They provide complex and indeterminate mechanisms to specific privacy and other policies for protecting access to personal information, and al-

low information to be shared that typically would not follow social and professional norms." Iannella (2009)

There have been numerous attempts to solve this problem in the past (Tonti *et al.* 2003) but none have been really successful, nor applicable to the Social Networks community. A new approach is required to manage seamless policy interaction for the Social Networks masses.

This raises four key challenges for policy languages:

- Policy Expression – how to unambiguously define the terms and conditions of a policy.
- Policy Transparency – how to ensure all parties are aware of the policy and its implications.
- Policy Conflict – how to detect potential incompatibilities between dependent policies.
- Policy Accountability – how to track policy exceptions and obligations.

With the emergence of Social Network interactions, the four policy challenges now need to be aligned with this new environment. Traditionally policy languages were designed based on a transaction environment. That is, the content and parties would enter into some explicit agreement under the control of some policy management system, for example, a DRM system buying music. However the Social Networks user base is more inclined to share content with friends and colleagues without any predetermined agreement. The focus has moved away from the transaction-based constrained policy (e.g., play the video 5 times over a 2 month period) to a regime based on sharing content to dynamic groups of people (e.g., any friend can comment on these photos.)

In this paper we first look at the Policy Expression Challenge through the emerging development of the Open Digital Rights Language (ODRL). We then look at a specific Use Case from Social Networks and apply the ODRL policy language to validate its expressiveness. We also investigate a computationally oriented approach for the formalisation and execution of policies. We then show the mapping between the formal model and ORDL. We conclude with a summary of the future research work and potential directions for policy challenges for Social Networks.

2 ODRL Version 2.0

The Open Digital Rights Language (ODRL) Version 2.0 Model (Guth and Iannella 2009b) has evolved over the past years from a specific rights management language to a more generic policy language. Figure 1 shows the core entities of the underlying information model. Like many other policy languages at the time, ODRL was modeled on the transaction-based policy environment. The new version of ODRL is motivated to broaden this scope to incorporate the unique needs of Social Network policies.

The key ideas of the ODRL model that are applicable to Social Networks include:

- A clear identification of the Asset (for any type of Social Network content).

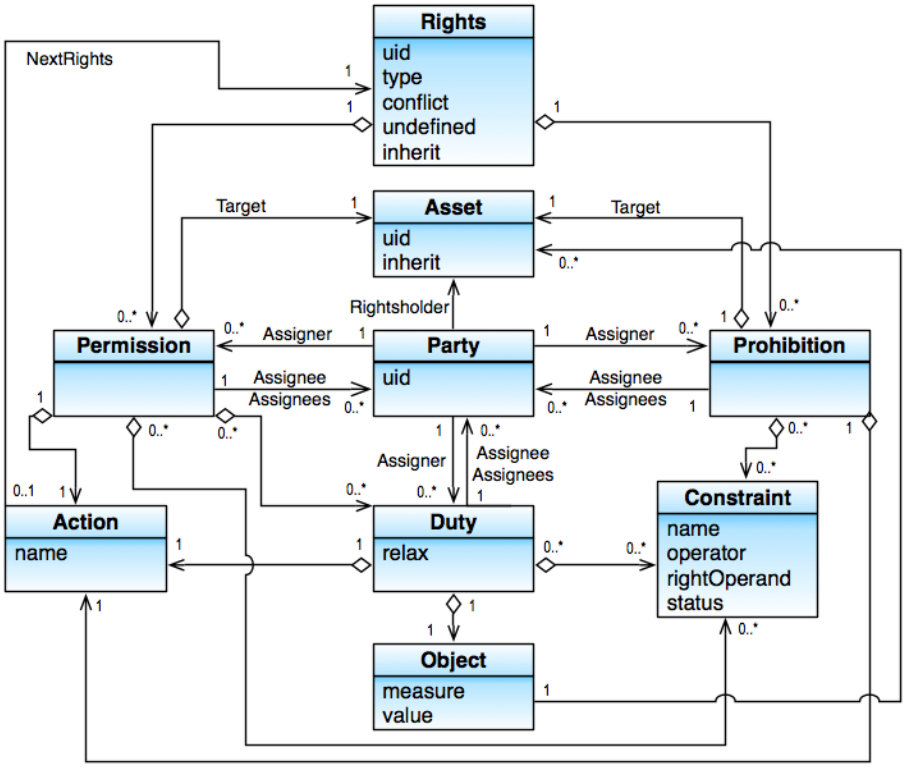

Figure 1: ODRL Version 2.0 Model

- Actions that are allowed to be performed (Permissions) or not allowed to be performed (Prohibitions) can be articulated.
- All the Parties involved can be specified (who assigns rights to whom).
- Any Duties on Parties can be stipulated (their obligations that must be meet).
- Constraints can be enumerated for any of the key entities.

The ODRL Version 2.0 Model is defined abstractly in UML. This was undertaken to ensure that the semantics of the policy language could standalone and was not dependent on any other underlying encoding model or syntax. Additionally, it focusses the work on Policy semantics and does not try and force the semantics into another framework.

The ODRL Version 2.0 Model will be implementable via both XML Schema and Semantic Web specifications but is still under development. However, it has reached the stage where it can be applied to different scenarios to validate its applicability to various communities.

3 Social Networks Requirements

We looked at two popular Social Networks (FaceBook and Flickr) and reviewed the types of conditions (or constraints) that can be applied to their content. Figure 2 shows examples of these conditions from these two Social Networks. The findings were also consistent with Professional Networks, such as LinkedIn and Plaxo.

What is clear from Figure 2 is that the policy decision points are focussed on constraining who the end user party is. That is, the content owner can specify these general types of limitations for who can access their content:

- Only the content owner (i.e., no one else)
- Specific (named) friends and colleagues (both allowed and not allowed)
- All direct friends or colleagues
- Your second level friends or colleagues (i.e., friends of friends)
- All Groups (that the content owner is a member of) or some Groups
- Everyone (i.e., public)

However, we believe that in a policy language for social networks should as general as possible, "As one cannot predict a priori what policies particular data owners might wish to impose, support for flexible and fine-grained models is essential" Simpson (2008).

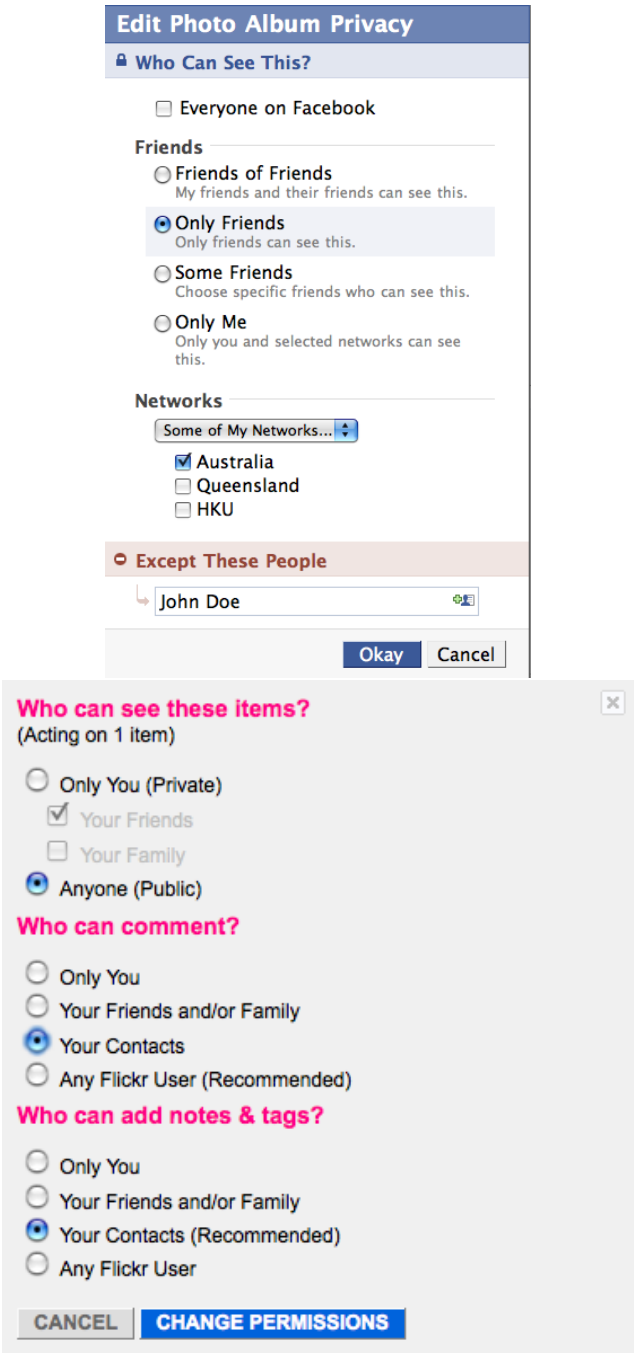

Figure 2: FaceBook (left) and Flickr (right) Privacy Settings

4 Social Networks Use Case

We looked at a specific use case (Passant *et al.* 2009) from the W3C Future of Social Networks Workshop (Jan 2009) in which:

> "Alice wants to give access to her wedding pictures only to people that are fellows on both Flickr and Twitter and that have a blog she commented at least twice during the last 10 days."

This use case also follows the similar theme in which the rights are bestowed on a constrained group of people. In this case, the group has additional constraints and requirements, specifically that Alice has commented on their Blog in the past 10 days. We then looked at expressing this use case in ODRL Version 2.0.

5 ODRL 2.0 Meets Web 2.0 (aka Social Networks)

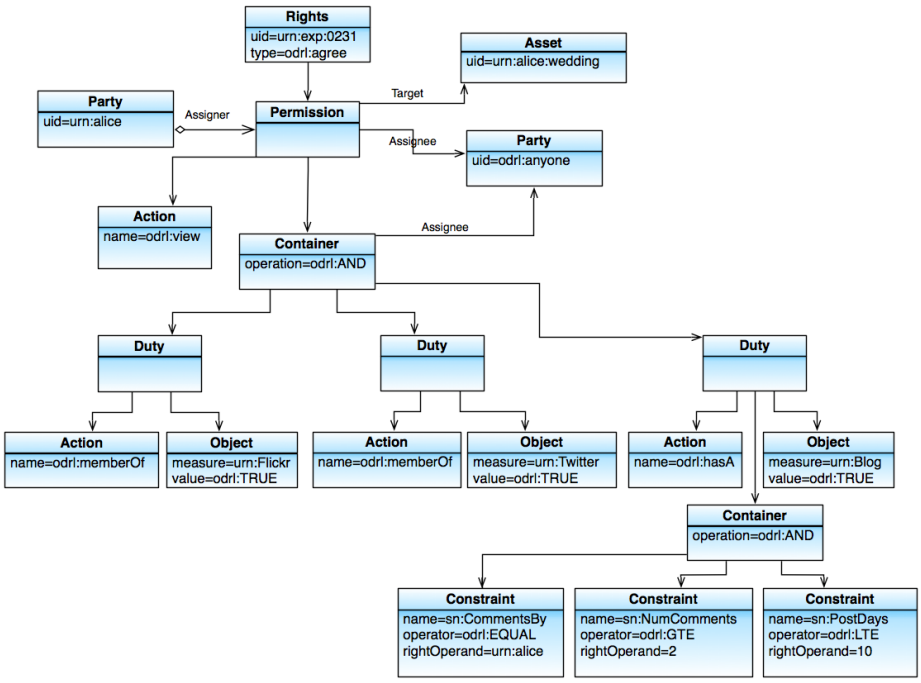

Figure 3: Alice Use Case in ODRL Version 2.0

Deconstructing the Alice Use Case leads to the following language expression needs:

- Identifying the Wedding Photos
- Alice is assigning rights

- The permission is viewing
- The recipient of the permission is the group of people that meet all of these criteria
  - Members of Flickr and Twitter, and
  - Have Blog sites, and
  - Alice has commented at least twice on these blogs,
  - In the last 10 days.

Figure 3 shows this instance of the Alice Use Case expressed in the ODRL Version 2.0 Model.

The instance shows the Wedding Photos Asset being the subject of a View Permission assigned by the Party Alice. The Assignee of the policy is the generic ODRL "anyone" Party that has three specific Duties that must be meet. These Duties then filter the "anyone" Party to the specific parties that match Alice's use case. In this case they must be members of Flickr and Twitter and have a Blog. The latter then has an additional constraint that Alice has commented on their blog in the past 10 days.

This mapping exercise of the Use Case has highlighted some additional requirements for the ODRL Version 2.0 evolution. Specifically, it has identified the need for a generic "anyone" party and membership semantics that need to be part of the ODRL Core Metadata (currently in Working Draft status).

6 ODRL 2.0 Meets Web 3.0 (aka Semantic Web)

ODRL Version 1.1 has been widely deployed primarily in XML (with over a billion mobile handsets supporting the OMA profile of the ODRL language). ODRL Version 2.0 will also support an RDF/XML binding to capitalize on the Semantic Web opportunities. The binding will require some hard decisions on how to best fit the ODRL model into the RDF Model. For example, below is an example RDF/XML encoding of Alice use case:

```
<odrl:Permission>
 <odrl:Asset rdf:resource="urn:alice:wedding"/>
 <odrl:Action rdf:resource="http://odrl.net/actions/view"/>
 <odrl:Assigner rdf:resource="urn:alice"/>
 <odrl:Assignee rdf:resource="urn:odrl:anyone"/>
 <odrl:Duty rdf:nodeID="D1"/>
 <odrl:Duty rdf:nodeID="D2"/>
 <odrl:Duty rdf:nodeID="D3"/>
</odrl:Permission>

<odrl:Duty rdf:nodeID="D1">
 <odrl:Action rdf:resource="http://odrl.net/actions/memberOf"/>
 <odrl:Object rdf:parseType="Resource">
     <rdf:value rdf:resource="urn:odrl:TRUE"/>
     <odrl:measure rdf:resource="urn:Flickr"/>
 </odrl:Object>
```

```
</odrl:Duty>

<odrl:Duty rdf:nodeID="D2">
 <odrl:Action rdf:resource="http://odrl.net/actions/memberOf"/>
 <odrl:Object rdf:parseType="Resource">
     <rdf:value rdf:resource="urn:odrl:TRUE"/>
     <odrl:measure rdf:resource="urn:Twitter"/>
 </odrl:Object>
</odrl:Duty>

<odrl:Duty rdf:nodeID="D3">
 <odrl:Action rdf:resource="http://odrl.net/actions/hasA"/>
 <odrl:Object rdf:parseType="Resource">
     <rdf:value rdf:resource="urn:odrl:TRUE"/>
     <odrl:measure rdf:resource="urn:Blog"/>
 </odrl:Object>
 <odrl:Constraint rdf:nodeID="C1"/>
 <odrl:Constraint rdf:nodeID="C2"/>
 <odrl:Constraint rdf:nodeID="C3"/>
</odrl:Duty>

<odrl:Constraint rdf:nodeID="C1">
 <odrl:name rdf:resource="urn:socialnet:commentsBy"/>
 <odrl:operator rdf:resource="urn:odrl:EQUAL"/>
 <odrl:rightOperand rdf:resource="urn:alice"/>
</odrl:Constraint>

<odrl:Constraint rdf:nodeID="C2">
 <odrl:name rdf:resource="urn:socialnet:numComments"/>
 <odrl:operator rdf:resource="urn:odrl:GTE"/>
 <odrl:rightOperand rdf:resource="2"/>
</odrl:Constraint>

<odrl:Constraint rdf:nodeID="C3">
 <odrl:name rdf:resource="urn:socialnet:postDays"/>
 <odrl:operator rdf:resource="urn:odrl:LTE"/>
 <odrl:rightOperand rdf:resource="10"/>
</odrl:Constraint>
```

Some encoding and mapping issues still need to be addressed such as the best fit for the ODRL Container model. For example, can the RDF Collection mechanism be utilised to express the boolean relationship with the three Constraints.

A major issue will be the mapping of the ODRL Permission/Prohibition model into RDF to infer policy conflicts. ODRL Version 2.0 has introduced a precedence mechanism to guide conflict detection. How this can be best modelled in RDF will be an interesting challenge in itself. We can imagine an extension to the Alice use case where she has also given permission for anyone who is a member of MySpace full access to all her photos. This will be in conflict with her original use case as now "all her photos" includes her wedding photos. How can we detect this and warn her of this conflict?

7 Computing Policies

In the previous sections we have examined some essential requirements for a policy language for social networks, and we have seen to what extent ODRL meets those requirements. However, a policy language has to be implemented to identify the properties enjoyed by resources and members in a network. In this section we provide a computationally oriented approach to this problem. In addition we examine some further aspects relevant to the deployment of executable specifications for policies in social networks.

Our proposal to address the issue of how to implement a policy language is based on FCL, a logical approach proposed by Governatori (2005) for the representation of an executable contract language and further proposed for the study of compliance of business processes (Governatori *et al.* 2006).

To illustrate some of the features needed we extend the Alice example.

Suppose that the network offers members the facility to create blacklists where a member can list members of the networks that cannot access the member resources, and the user can specify restrictions on the resources available to members in a blacklist. Alice decided that blacklisted members cannot access her resources at all. Moreover, suppose that Alice put Bob in the photo blacklist, but she has posted a few recent comments on Bob's blog, and Bob is a member of the categories listed in Alice conditions to access her wedding pictures.

Consider another example: The network has another feature. Each user has a profile page, and the user has to upload a picture to the profile page, and this picture is available to everybody in the network. Members who do not comply with the above conditions cannot access other members' private resources.

Alice puts a picture of her wedding as her public photo. Carl is another Flickr and Twitter fellow of Alice (not in her blacklist, and she repeatedly posted in his blog during the past week) who does not have his public picture in his profile.

The above examples illustrated some important features we are faced with when we want to implement a policy language in social networks (and not only in social networks).

- policy conditions have a normative nature;
- policy conditions can have exceptions;
- conditions in policy can conflict with each other;
- policies in a social networks can come from different sources;
- policy conditions sometimes involve violations of other policy conditions.

7.1 Executable Policy Specifications

We briefly present the basic of FCL (Formal Contract Logic). FCL results from the combination of an efficient rule base non-monotonic logic formalism (Defeasible Logic (Antoniou *et al.* 2001, 2006)) and a Deontic Logic of violations (Governatori and Rotolo 2006). Deontic Logic is the branch of logic studying normative like concepts like

obligations, permissions, prohibitions, ... These notions correspond to the right, duty, prohibition notions illustrated in the previous sections. Deontic logic extends first order logic with the deontic operators O, P and F denoting obligations, permissions and prohibitions. The deontic operators satisfy the following equivalence relations:

$$OA \equiv \neg P \neg A \quad \neg O \neg A \equiv PA \quad O \neg A \equiv FA \quad \neg PA \equiv FA.$$

The operators also satisfy the following relationship $OA \rightarrow PA$, meaning that if A is obligatory, then A is permitted. This relationship can be used to ensure checking of the internal consistency of the obligations in a set of norms, i.e., whether it is possible to execute obligations without doing something that is forbidden. FCL then extend deontic logic by considering directed deontic operators. This means that each operator can be indexed by the subject and the beneficiary of the normative concept. Thus for example $O_s^b A$ means that s has the obligation of A with respect to b, where A could be the statement "prevent disclosure of personal information". Similarly $F_s B$ means that B is forbidden for s (where B, for example, means "access private data"). The deontic logic component of FCL gives us the ability to handle the normative aspects of ODRL, as well as the aspect of how to handle violations. Often the treatment of violations is not properly addressed in other deontic logics (see Carmo and Jones (2002) for a detailed presentation of the problems related to violations in deontic logic).

Typically normative systems, of which policies are particular instances, include conditions that are activated to compensate breaches of other conditions. To capture this aspect we introduce the *reparation* (or compensation) operator $\otimes$, to be used in expression like $OA \otimes OB$. The meaning of an expression like $OA \otimes OB$ is that we have the obligation of A (i.e., OA), but in case this is violated, i.e., we have the negation of A, i.e., $\neg A$, then the obligation OB is in force. This means that achieving B compensate for failing to fulfill the obligation OA. Since the compensation is an obligation as well, it is possible that it is violated, and so it can trigger an additional compensation. To accommodate this we allow chains of obligation compensation of any length. Thus, for example we can have expressions like

$$OA_1 \otimes \cdots \otimes OA_n$$

(also called obligations chains or simply chains) saying that the main obligation is OA_1 but in case this is violated, then the next obligation is OA_2, and even if this is violated, then we trigger the obligation OA_3 and so on. In these chains we can have obligations and prohibitions[1]. Permissions can appear only as the last element of a chain. The reason for this is that it is not possible to have a violation of a permission, and so it is meaningless to have a compensation for something that cannot be violated.

The defeasible logic component of FCL allows us to capture exceptions and conflicts, more specifically the inference mechanism of FCL is an extension of Defeasible Logic.

Defeasible logic, originally created by Donald Nute (1994) with a particular concern about efficiency and implementation, is a simple and efficient rule based nonmonotonic formalism. Over the years, the logic has been developed and extended, and

[1]Please remember that prohibitions can be represented as obligations, $FA \equiv O \neg A$.

several variants have been proposed to model different aspects of normative reasoning and it encompasses other formalisms for normative reasoning.

The main intuition of the logic is to be able to derive "plausible" conclusions from partial and sometimes conflicting information. Conclusions are *tentative* conclusions in the sense that a conclusion can be withdrawn when we have new pieces of information.[2]

The knowledge in a Defeasible Theory is organised in *facts* and *rules* and *superiority relation*.

- Facts are indisputable statements.
- Defeasible rules are rules that can be defeated by contrary evidence.
- The superiority relation is a binary relation defined over the set of rules. The superiority relation determines the relative strength of two (conflicting) rules.

The meaning of a defeasible rule, like

$$A_1, \ldots, A_n \Rightarrow C$$

is that normally we are allowed to derive C given $A_1, \ldots, A_n$, unless we have some reasons to support the opposite conclusion (i.e., we have a rule like $B_1, \ldots, B_m \Rightarrow \neg C$).

Defeasible Logic is a "skeptical" non-monotonic logic, meaning that it does not support contradictory conclusions. Instead, Defeasible Logic seeks to resolve conflicts. In cases where there is some support for concluding A but also support for concluding $\neg A$, Defeasible Logic does not conclude either of them (thus the name skeptical). If the support for A has priority over the support for $\neg A$ then A is concluded.

A defeasible conclusion is a tentative conclusion that might be withdrawn by new pieces of information, or in other terms it is the 'best' conclusion we can reach with the given information. In addition, the logic is able to tell whether a conclusion is or is not provable. Thus, it is possible to have the following two types of conclusions:

- Positive defeasible conclusions: meaning that the conclusions can be defeasible proved;
- Negative defeasible conclusions: meaning that one can show that the conclusion is not even defeasibly provable.

A (positive) defeasible conclusion A can be derived if there is a rule whose conclusion is A, 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 $\neg A$ (the negation of A) has prerequisites that fail to be derived. In other words, a conclusion A is (defeasibly) derivable when:

1. A is a fact; or
2. there is an applicable defeasible rule for A, and either
   (a) all the rules for $\neg A$ are discarded (i.e., not applicable) or

[2]For a full presentation of the logic, refer to (Antoniou *et al.* 2001, 2006)

(b) every applicable rule for $\neg A$ is weaker than an applicable strict or defeasible rule for A.

A rule is applicable if all elements in the body of the rule are derivable (i.e., all the premises are positively provable), and a rule is discarded if at least one of the elements of the body is not provable (or it is a negative defeasible conclusion).

The main difference between Defeasible logic and FCL is that in FCL the conclusion of a rule is an obligation chain (possibly a trivial chain where we have only one element).

Accordingly the reasoning mechanism to derive conclusion is an extension of that for defeasible logic. In defeasible logic the conclusion of a rule is a single literal and not a reparation chain. Thus, the condition that OA appears in the conclusion of a rule means in defeasible logic that OA is the conclusions of the rule. FCL extends defeasible logic with reparation chains, thus, we have to extend the reasoning mechanism of defeasible logic to accommodate the additional construction provided by FCL. To prove OA, we have to consider all rules with a reparation chain for OA, where for all elements before OA in the chain, the negation of the element is already provable. Thus to prove A given a rule

$$P_1, \ldots, P_n \Rightarrow OC_1 \otimes \cdots \otimes OC_m \otimes OA \otimes OD_1 \otimes \cdots \otimes OD_k,$$

we have that $P_1, \ldots, P_n$ must be all provable, and so must be $\neg C_1, \ldots, \neg C_m$. For the full details see Governatori (2005).

7.2 FCL at Work for Social Networks

In this section we will examine how the feature of FCL presented above address some important aspects of social network policies.

7.2.1 Exceptions

The superiority relation can be used to model exceptions. An exception is a situation where we have some specific information that prevents the derivation of the otherwise normal conclusion. Consider the following policy for handling accessing resources in a social network: 'Member resources on the network can be access by everybody, unless a resource is declared private'.

This condition can be expressed by the default rule

$$r_1 : resource(x) \Rightarrow P\, access(x)$$

The predicate *resource* states that x is a resource in the network, and then, the rule states that if something is a resource then access is permitted to it. In this case we can use an undirected permission to represent the condition (everybody in the network can access the resource, thus there is no need to specify a specific beneficiary for it). To capture the *unless* part of the policy, expressing the exception, we can use the rule

$$r_2 : private(x), \neg owner(x,y) \Rightarrow O_y \neg access(x)$$

The meaning of this rule is that if something is a private resource and somebody (y) is not the owner of the resources, then it is forbidden to y to access the resource. Remember that a prohibition is an obligation followed by a negation, i.e., $O\neg$.

In case both rules apply, then we cannot conclude anything, because the two rules are in conflict, something is at the same time permitted and prohibited. Thus to express that we have an exception we have to specify the superiority relation between the two rules. In this case, we can specify that $r_1 \prec r_2$, making thus r_2 stronger than r_1. This means that in case both fire, r_2 takes precedence over r_1, and we can derive the conclusion of r_1. Accordingly we can deny access to private resources.

7.2.2 Conflicts

In a social network we can have policies from multiple sources, for example we can have multiple policies from networks in a social network aggregator as well as policies from individual members. As result it is possible that policies are in conflict with other policies.

Let us go back to the Alice scenario. Her rule about access to her wedding pictures can be formalised as follows;

$$p_1 : wedding\_photo(x), flickr(y), twitter(y), blog(z,y), posted(a,z,t_1), posted(a,z,t_2), \\ t_1 > Now - 10, t_2 > Now - 10 \Rightarrow P_y access(x) \quad (1)$$

The predicates in the rule mean, respectively: x is the id of a wedding photo, y is a friend of Alice in Flickr, y is a friend of Alice in Twitter, z is the blog of y, Alice posted to the z's blog at time t_1, and at time t_2, and t_1 and t_2 are less than ten days ago (we assume that *Now* returns the current time in days). Then the rule says that if all the conditions above are satisfied then y is permitted to access the picture whose id is x.

Let us suppose now that x is a private wedding picture and that the other conditions in the wedding rule p_1 are satisfied. Given the two rules we have a possible conflict between the two rules. According to rule p_1 access should be granted. On the other hand, the restriction on private resources, rule r_2, prevents granting the access to the picture.

Again to solve the issue we have to use the superiority relation. Here one has to set that $r_1 \prec p_1$.

Notice that conflicts can arise both from rules from different sources, but, as in the case of exceptions, they can originate from one and the same source, as the following rule illustrate.

Consider the policy about blacklisting 'if a member is a in blacklist then the member cannot access private resources'. This rule can be expressed as follows:

$$p_2 : private(x), blacklist(y) \Rightarrow O_y \neg access(x)$$

Now to be effective this rule must be at least as strong as the rule to grant access (i.e, rule p_1). Thus we further need that $p_2 \prec p_1$.

7.2.3 Policies for Violations

The conditions that 'each member has to upload a picture to the profile page, and this picture is available to everybody in the network' and 'members who do not comply with the above conditions cannot access other members' private resources', can be expressed by the rule:

$$r_3 : \Rightarrow O_x publish\_public \otimes O_x \neg access(y)$$

This rule establishes that a member has the obligation to publish at least one picture (visible to everybody), otherwise, the member cannot access any private resource. Accordingly, in case the first condition, i.e., $O_x publish\_public$, is violated, meaning that we have $\neg publish\_public$ (meaning that there are no public pictures). Then we can trigger the compensation (the sanction preventing the member to access other's member private resources). Notice that then the rule is in conflict with r_1 and p_1. So to make it effective we have to specify that $r_3 \succ r_1$ and $r_3 \succ p_1$.

7.3 Discussion

The reasoning mechanism presented above is to determine the obligations, permissions and prohibitions in force for a particular situation. The mechanism is based on the proof conditions of defeasible logic, which offers a constructive proof theory (Antoniou *et al.* 2001, 2006). The important aspect is that the constructive proof theory allows us to look at the derivation of a conclusion and to see the rules and facts used to derive a conclusion, and thus we can provide a full justification of why we obtained a specific outcome (Antoniou *et al.* 2008). This feature caters for the accountability of our approach to policies.

Closely related is the efficiency and scalability. We envision that a policy server for a social network should be able to handle a large amount of requests. Thus the efficiency of the reasoning mechanism is of paramount importance. The outcome of an FCL policy can be computed in linear time (Maher 2001). Efficient implementations exist, and the most recent implementations fully support Semantic Web standards (RDF, RuleML) (Bassiliades *et al.* 2006), and extensions to deal with normative conditions have been proposed (Governatori 2005).[3]

Defeasible logic and FCL have been proposed and applied for the representation of different aspects of normative reasoning (modelling contracts (Governatori 2005), modelling complex normative notions such as normative power and delegation (Governatori and Rotolo 2008), modelling norm changes (Governatori *et al.* 2007b), and business process compliance (Governatori *et al.* 2006, Sadiq *et al.* 2007)). Thus the principles on which FCL is based on seem to offer a faithful and conceptual representation of normative concepts such as those required to model policies in social networks. Therefore FCL offers a transparent framework for the domain under analysis.

In addition to the reasoning mechanism to derive the normative requirements in force, FCL has other reasoning mechanisms. For example, it is equipped with mechanisms to generate normal forms for a set of rules (Governatori and Milosevic 2006).

[3] A Java based open source of Defeasible logic and FCL is available at `http://spin.nicta.org.au/demo/SPINdle/`, see Lam and Governatori (2009).

The normal form makes explicit all rules that can be obtained by combining given rules, and then removes redundant rules, i.e., rules whose meaning is included in the meaning of other rules. The generation of a normal form has several advantages for the design of policies. Since all conditions are made explicit, then it is possible to identify conflicts between policy conditions, and at the same time it is possible to have a complete picture for the terms and conditions of a policy (Governatori and Milosevic 2006). Accordingly, normal forms accounts for both the transparency and coverage of a policy. Furthermore the analysis of the normal form of a policy can be used to determine whether a particular effects is achieved given a particular situation. This can be also simulated by testing the a case by running a FCL theory based on some generated data provided by an user.

8 Mappings between ORDL and FCL

We briefly discuss the mapping between ODRL and FCL, in particular we examine the correspondence between classes in ODRL and elements of FCL. In FCL we have three categories of objects: propositions, deontic operators and rules. Atomic propositions are built from predicates plus terms (where a term is either a variable or a constant), where variables and constants denote elements of the domain. For ODRL and social networks the domain of individuals consists of the resources and members of the social networks. Thus we can establish a mapping between the ODRL classes **Asset** and **Party** and individuals in FCL. Predicates describe properties of element of the domain and relationship between them, thus an FCL theory modelling the policy of a social network has to provide predicates corresponding to the attributes of the classes and to the relationships between classes. Thus for example the *Rightholder* relationship between the classes **Asset** and **Party** is modelled by the predicate $owner(x,y)$. In addition we create a predicate for each instance of the class **Action**, and we create appropriate predicates for instances of the class **Constraint**.

The classes **Permission**, **Duty**, and **Prohibition** are mapped to the deontic operators P and O (and $O^{\neg}$ for instances of **Prohibition**). As we have seen in Section 7 the deontic operators bound literals (the literals should be a literal corresponding to one action, though in general in FCL a deontic operator can bind in general any type of proposition), and can be indexed by subject and beneficiary. These correspond to assignee and assignor in ODRL.

Finally a rule is a relationship between a set of instances of the class **Constraint** and an instance of any of the classes **Permission**, **Obligation** and **Prohibition**.

9 Richer Policy Languages

In the previous sections we restricted ourselves a version of FCL that could be mapped directly to the current version of ODRL. In the rest of the section we examine how to enrich our framework, and we will do it taking a computationally oriented approach. This mean that we will consider features for which (efficient) computation methods

have been proposed[4]. Specifically, here we concentrate in adding constructions to handle temporal aspects. In addition, we enrich our framework with a rich ontology of obligations. However, before moving to further extensions we have to see how some features of FCL not discussed in the mapping given in the previous section can be handled by ORDL.

9.1 Extending ORDL

In Section 7 we have seen that FCL can handle violations, using the $\otimes$ operator, and the superiority relation ($\prec$), but these do not have counterparts in ODRL. This, however, is not a real limitation. As it was discussed by Governatori (2005) the violation operator is useful for the design of the policy (i.e., when using the normal form to investigate the characteristics of a policy), but it is not required for the computation of the requirements. Indeed a rule $A \Rightarrow OB \otimes OC$ is equivalent to the following two rules $A \Rightarrow OB$ and $A, \neg B \Rightarrow OC$.

For the superiority relation, Antoniou *et al.* (2001) show how to take a set of rule with superiority theory in Defeasible Logic and transform it into an equivalent theory without superiority relation.

While the two constructors above are not required to specify the rule comprising a policy, we suggest that ODRL is extended to include elements corresponding to such notions to improve how policies are defined, relieving policy authors (where for social networks, for a private profile the designer can be the user itself) for the burned of specifying how to implement her policy conditions, and thus resulting in a more conceptual language for policies.

9.1.1 Override When Conflict

ODRL Version 2.0 supports the "conflict" attribute (at the parent rights element level) to resolve conflicts arising from the merging of licenses, specifically when there are conflicting actions in the Permissions and Prohibitions. The range of values are:

- perm: the Permissions will always takes precedence;
- prohibit: the Prohibitions will always takes precedence;
- invalid: the license is not valid (the default).

The primary focus of this feature is to delineate between priorities between Permissions and Prohibitions, and not within these groups.

To support the conflict scenarios described in Section 7.2.2 ODRL would need to support a more fine grained approach to conflict resolution.

One solution would be to incorporate a precedence attribute deeper into the rights expression language structure. For example, consider the scenario in Figure 3 such

[4]The rationale behind this choice should be obvious; while it is possible to extend a policy language with some representation features in the language, this exercise would be pointless unless these features could be implemented for the execution of a policy; on the other hand, if a feature has been given in a computationally oriented logical framework, then a language for it must already exist, and thus the issue is how to port it to a specific policy language.

that the three Duties need to be resolved for potential conflict. The three Duty entities could be expressed with a "precedence" attribute indicating the superiority relationship, as each Duty entity can be uniquely identified.

In RDF/XML, this may be represented as:

```
<odrl:Duty rdf:nodeID="memberFickr">
<odrl:Duty rdf:nodeID="memberTwitter">
  <odrl:precedence rdf:nodeID="memberFlickr">
  ...
</odrl:Duty>
```

In this example, the memberTwitter Duty takes a greater precedence over the memberFlickr Duty.

Such a new structure would also only may sense within a Container element, as the members would then be semantically typed.

9.1.2 Penalty for Duty

When assigning a Duty in ODRL Version 2.0, the language assumes that the recipient will either perform the Duty (and then get access to the asset) or not perform the Duty (and hence, not get access to the asset). There is one exception in that Duty entity may contain the "relax" boolean that indicates if the duty may be fulfilled at anytime, including after the Permission has been utilised by the Assignee. The default Relax boolean setting for all Duty entities is false meaning that the Duties must be fulfilled before the rights can be exercised. If the value is false then the Duty can be fulfilled at anytime, but still must be fulfilled at some point in time in the future.

However, there is no concept of "compensation" for not performing the Duty, as described in Section 7.2.3 In the current case, if you did not perform the Duty (with or without the relax" boolean) there are no other obligations on you.

It would be possible to support such a concept by extending the ODRL Model to migrate the "relax" boolean into a "violation" attribute. This attribute would include the semantics of:

- mandatory: the Duty must be performed
- relax: the Duty may be performed (same as the relax = true)
- failure: the Action that must be performed if the Duty cannot

However, a recursive filter will need to be supported (in implementations) to ensure that the failure Action is not the same as the original Duty action.

9.2 Time-based Policies

Time-based policies are an important part of social network policies and a key provenance factor when trying to compute conflicts between disparate policies. For example, a policy may only be valid for one year and only overlap another policy by a few months. Hence it is important to determine that, in the future, the policies may not conflict, but may do today.

9.2.1 Adding Time to ODRL

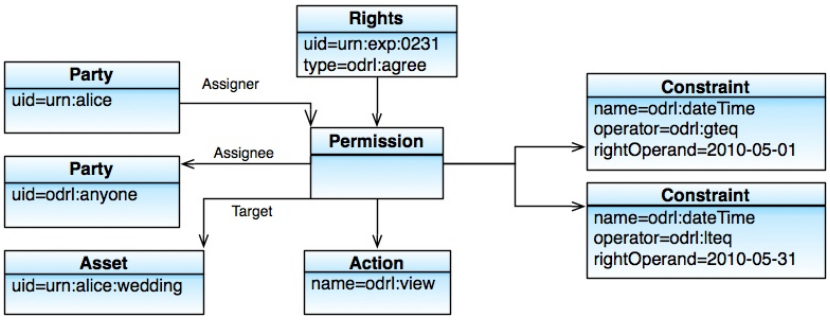

Figure 4: Adding time to ODRL normative elements

The ODRL Common Vocabulary (Guth and Iannella 2009a) details all the applicable Actions and Constraints, including the "dateTim" term to support date and time-based constraints. These are typically directed towards specific Permissions, such as View a photo between 1st May to 31st of May 2010.

However, the ODRL Version 2.0 Model was deliberately developed to support constraints at the individual Permission/Prohibition level only. The previous ODRL Version 1.1 (Iannella 2002) allowed a Context element with a date range applicable to the entire expression. This was a useful shortcut but made formal modelling more difficult.

The current solution would be to attach the date constraints to each Permission and/or Prohibition in the expression. This is not onerous as the same date constraint entity can be linked to multiple Permissions/Prohibitions. Figure 4 shows and expanded version of Figure 3 with a date constraint.

9.2.2 Adding Time to FCL

For the extension of FCL with time we follow the proposal by Governatori *et al.* (2005). One of the main ideas of this approach is that each proposition has attached to it a timestamp indicating when the proposition holds, accordingly the language of FCL is extended to handle basic expression of the type p^t, where p is a proposition and t is a timestamp. Thus for example the expression $posted(Alice, \texttt{bob.example.com/blog})^{20100101}$ means that Alice posted on Bob's blog at `bob.example.com/blog` on January 1st, 2010.

The second key concept is that conclusions can be asserted either in a persistent way or in a transient way. This is achieved by having two classes of rules, persistent and transient rules. A persistent rule has the form:

$$a_1^{t_1}, a_2^{t_2}, \ldots, a_n^{t_n} \Rightarrow^{\pi} c^t$$

The meaning of the above rule is that if a_1 holds at time t_1, and a_2 holds at time t_2, and so on for all elements of the antecedent of the rule, then we can assert that c holds at time t, and it continues to hold until it is terminated.

A transient rule is an expression:

$$a_1^{t_1}, a_2^{t_2}, \ldots, a_n^{t_n} \Rightarrow^{\tau} c^t$$

The interpretation is similar to that of persistent rules with the difference that the conclusion c is guaranteed to hold only at time t (and not after it).

The distinction between the two types of rules allows us to capture the distinction between the following two conditions:

> "current friends in Flickr can access wedding photos"

and

> "friends in Flicker can access wedding photos"

Both are represented by a rule

$$\mathit{flickr}(y)^t, \mathit{wedding\_photo}(x)^t \Rightarrow^X P_y\mathit{access}(x)^t$$

but in the first case we have to set X to τ to establish that the rule is a transient rule. This means that the conditions y is a friend in Flickr, the photo x is classified as a wedding photo are determined at the same time when the resource is accessed by y. If we set X to π, i.e., to a persistent rule, then once the conditions in the antecedent are satisfied once, then access to the picture is granted, and y can access the picture at a time following the time the access has been granted (unless the privilege has been revoked).

Based on this discussion we the question is how can we determine if a proposition holds at a specific time. This means that to determine whether a proposition p holds at time t, we can check that it hold transiently (we can use a transient rule to prove the conclusion at that time) or it was proved persistently at a previous time (or at that time). The reasoning mechanism for transient conclusions is the same as FCL without timestamps, while we have to extend the mechanism to capture temporal persistence. Thus to prove that the proposition was proved using a persistent at a time t' preceding the current time t, and that for all instants t'' between t' and t, the proposition p was not terminated. This amounts to check that every rule for the negation of the proposition leading to concluding it between t' and t'' were not applicable, i.e., some of the premises did not hold, or the rule was weaker that an applicable rule for p. For the full details see (Governatori *et al.* 2005, Governatori and Rotolo 2010).

The temporal framework outlined above enjoys several interesting features: Governatori and Rotolo (2010) show that the set of conclusions can be computed in time linear to the size of the theory (thus extending the same complexity result of FCL without timestamps). The logic with the temporal extension has been implemented (Rubino and Rotolo 2009) and both the logic and the implementation have been tested and validated. Specifically the validation was done by encoding the Road Traffic Restriction Regulation of the Town of Piacenza (Italy) (Governatori *et al.* 2010). The experiments have shown that the implementation handles large theories, the proposed framework offers sped-ups over alternative encodings, the logic is able to model and reason with typical of policies and norms. A feature that is particularly interesting for social network policies is the ability to represent deadlines.

In the rest of this section we illustrate how the logic at hand can model deadlines. An alternative analysis has been developed by Governatori *et al.* (2007a), but the logic used there was based on introducing in FCL temporal intervals, thus posing complexity limitations. Here, we show that a more efficient variant of FCL can do the same job. The idea of deadline refers to the notion of obligation, which is parametrized by temporal instants. Consider the following example.

(1) New members must complete their profile within 30 days of the date in which they have joined the network

The condition above states an obligation for to complete the member profile within 30 days upon the opening of an account.

We can distinguish *achievement obligations*, like policy (1), from *maintenance obligations*, like policy (2) below. For an achievement obligation, a certain condition must occur at least once before the deadline. For maintenance obligations, a certain condition must obtain during all instants before the deadline. Consider the following example.

(2) Member must keep a public picture, for 30 days after opening an account.

In (2), the deadline only signals that the obligation is terminated. A violation occurs when the obliged state does not obtain at some point before the deadline.

Policy (1) can be represented as follows. The deadline refers to an obligation triggered by the opening of an account (open_{init}): such an obligation is persistent. After that the member is obliged to complete the profile. The obligation terminates only when it is complied with (open_{term}). Note that the obligation itself may even persist after the deadline (like in policy (1)). Generally, a deadline signals that a violation of the obligation has occurred (rule open_{viol}).[5]

$$\begin{array}{ll}
\text{open}_{init} & \text{open_account}(x)^{t_1} \Rightarrow^{\pi} O_x\text{complete_profile}(x)^{t_1} \\
\text{open}_{term} & O_x\text{complete_profile}(x)^{t_2}, \text{complete_profile}(x)^{t_2} \Rightarrow^{\tau} \neg O_x\text{complete_profile}(x)^{t_2+1} \\
\text{open}_{viol} & \text{open_account}(x)^{t_1}, O_x\text{complete_profile}(x)^{t_1+30} \Rightarrow^{\tau} \text{viol}(\text{open})^{t_1+30}
\end{array}$$

Suppose that the set of facts is $\{\text{open\_account}(a)^1, \text{complete\_profile}(a)^{20}\}$. At time 1 we can derive $\text{open\_account}(a)$, which makes rule open_{init} applicable, leading to conclude Opay at time 1 (persitently), that is, an obligation to complete the profile applies persistently. Rule open_{term} terminates the obligation at 21. Therefore rule open_{viol} is not applicable, and we cannot derive a violation: $\text{viol}(\text{open})$ at time 30.

Policy (2) can be represented as follows.

$$\begin{array}{ll}
\text{keep}_{init} & \text{open_account}(x)^{t_1} \Rightarrow^{\pi} O_x\text{keep_public_photo}^{t_1} \\
\text{keep}_{term} & \text{open_account}^{t_1} \Rightarrow^{\tau} \neg O_x\text{keep_public_photo}^{t_1+30} \\
\text{keep}_{viol} & O\text{keep_public_photo}^{t_2}, \neg\text{keep_public_photo}^{t_2} \Rightarrow^{\tau} \text{viol}(\text{keep})^{t_2}
\end{array}$$

[5] In general we can combine the initiation rule and the violation rule into a single rule using the $\otimes$ operator as we did in the previous sections. However, to better illustrate the idea and the structure of deadlines we prefer to keep the two rules separated.

Notice that we may have other examples where a maintenance obligation holds indefinitely (and not only for 30 days). In those cases, the clause keep_{term} is not needed and no termination is required.

The examples above specify a 30-days period, but in general we could have any action or event, such as 'having dinner', in 'I must finish work before dinner', when you do not exactly know when dinner will be. In those cases, we will have to add an extra premise (representing that action or event) in the body of violation clause.

By definition, maintenance obligations do not persist after the deadline. But achievement obligations often do persist, until they are achieved. However, this is not the case for all achievement obligations. It is possible to have achievement obligation that are terminated by the occurrence of the deadline. For more details see Governatori *et al.* (2007a).

10 Related Work

While there has been debate about policies (e.g., access control and privacy policies) for social networks, the research on how to express policies for social networks and on how to reason with and on has been by large neglected. The only notable exception is about access control policies. In general the main idea of these proposals is to define a notion of trust based on the degree of relationships among members of social networks based on the typology of the network.

Ali *et al.* (2007) adopt a multi-level security approach, where trust is the only parameter used to determine the security level of both users and resources.

Kruk *et al.* (2006) propose an extension of Friend of a Friend (FOAF), called D-FOAF for an ontology-based identity management system for social networks, where access rights and trust delegation management are provided as additional services. The key point is that relationships are associated with a trust level, which denotes the level of 'friendship' existing between the users participating in a given relationship.

The work by Carminati *et al.* (2009b) discusses a semi-decentralized discretionary access control model and a related enforcement mechanism for controlled sharing of information in Web Based Social Networks is presented. The model allows the specification of access rules for online resources, where authorized users are denoted in terms of the relationship type, depth, and trust level existing between nodes in the network. Policies are limited to access control of resources, and users who want to access some resources have to provide credentials.

The proposal of Carminati *et al.* (2009a) consists of using semantic web technology to describe the various components involved in access control for social networks, namely it proposes ontologies for: personal information, personal relationships, resources, user/resources relationships, and actions. Furthermore it uses an ontology to manage three aspects: access control authorization, admin authorization and prohibitions. Security rules are then implemented in SWRL.

In general the main difference between the approaches just discussed and what we presented in this paper is that we do not restrict ourselves to access control policy. For example, we shown how to encode in our languages general policies for social networks. In addition we provide a rich ontology of obligations and we illustrated

how to model them. In this regard we would like to point out that Defeasible Logic has been discussed as formalism for access control related to XACML (Kolovski *et al.* 2007) and composition of secutiry policies (Lee *et al.* 2006). It is now recognised in the field of security and access control that deontic notions are important for proper modelling and verifying security conditions (Ni *et al.* 2008, Abadi 2008). Therefore, FCL combining both seems to be provide a good ground for this type of applications. In addition, reasoners for FCL (Lam and Governatori 2009, Governatori and Pham 2009) are compatible with Semantic Web data (RDF) and to some extent con be integrated with OWL ontologies (Governatori 2004).

11 Future Work

The work to date has focussed on developing an extensible Policy Language with the ODRL Version 2.0 Model and the Policy Expression challenge. There is still work to be performed on validating ODRL's expressibility and we plan to use new Social Network-based use cases from the FP7 PrimeLife Project (Bournez and Neven 2008). We will also work on expressing the ODRL Model in RDF/XML to support Semantic Web applications of the ODRL policy language.

Next on the list is the Policy Transparency challenge and we plan to investigate mechanisms to best inform Social Network users on the terms of policies that apply to the content they are using. The Policy Conflict challenge will be the most formidable as it will push ontology matching to the extreme, but is where the Semantic Web technologies will provide the most benefit. For example, in the Alice use case, if one of the potential assignees has a privacy policy on their blog account that suppresses key information (e.g., the number of postings) then how can we deal with the inconsistency with the original policy?

In this paper have assumed that the terms of the various policies are drawn from a common ontology. However, it is possible that policies use concepts from different sources and these sources adopt different ontologies to describe the terms of the (sub)policies. This is another case where we have to use semantic web technology. The address this issue, one has either to use techniques to align the different ontologies, or to assume a top-ontology for the domain of applications. We refrain here to further dwelling this aspects, since the policy languages we have adopted in this work, ODRL and FCL, are agnostic to this aspect. They assume basic languages describing the resources, personal information and actions for the domain understanding them as simply as logical proposition or predicates and build on them, thus it is not relevant whether these are represented in RDF, OWL or in other languages.

Finally, the Policy Accountability challenge will bring the most acceptance of policies in Social Networks as we move away from the traditional enforcement regimes and become consistent with society norms. Such as allowing for freedoms on Social Networks but making sure the user is aware of the consequences of their actions (or inactions). We don't wish to preclude the possibility of policy violations and at the same time provide a trusted experience for all Social Network users. A key challenge for all the community. The implication for a move towards Accountability is that all actions need to be recorded for later analysis. Ideally such auditing would be via trusted third

parties to allow for greater transparency and reduce the risk of any conflicts of interest.

12 Conclusion

Social Networks provide a unique circumstance now for the Policy and Semantic Web communities to apply previous research outcomes and implementation experiences at a scale not yet seen. The results will be significant but must address the needs of users first otherwise the uptake of these technologies will be a lost opportunity. We have shown some promising beginnings with a widely-deployed rights language that has evolved to become a general policy language. The future extensions and iterations of the language, and supporting systems, will rise to meet the challenges of the burgeoning policy-oriented semantic web.

Acknowledgements

This paper is an extended and revised version of Governatori and Iannella (2009). We would like to thank the anonymous EDOC 2009 referees for their valuable comments.

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program and the Queensland Government.

References

Abadi, M., 2008. Variations in Access Control Logic. *In*: R. van der Meyden and L. van der Torre, eds. *9th International Conference on Deontic Logic in Computer Science (DEON 2008)*, Vol. 5076 of *Lecture Notes in Computer Science* Springer, 96–109.

Ali, B., Villegas, W., and Maheswaran, M., 2007. A trust based approach for protecting user data in social networks. *In*: K. Lyons and C. Couturier, eds. *Conference of the Centre for Advanced Studies on Collaborative Research (CASCON)* IBM, 288–293.

Antoniou, G., *et al.*, 2008. Proof explanation for a nonmonotonic Semantic Web rules language. *Data & Knowledge Engineering*, 64 (3), 662–687.

Antoniou, G., Billington, D., Governatori, G. and Maher M.J., 2001. Representation Results for Defeasible Logic. *ACM Transactions on Computational Logic*, 2 (2), 255–287.

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

Bassiliades, N., Antoniou, G., and Vlahavas, I., 2006. A Defeasible Logic Reasoner for the Semantic Web. *International Journal on Semantic Web and Information Systems*, 2, 1–41.

Bournez, C. and Neven, G., 2008. Draft Requirements for Next Generation Policies. PrimeLife Deliverable H5.1.1, 11 December 2008. [online] `http://www.primelife.eu/images/stories/deliverables/h5.1.1-policy_requirements-public.pdf` [5 August 2010].

Breslin, J.G. and Decker, S., 2007. The Future of Social Networks on the Internet: The Need for Semantics. *IEEE Internet Computing*, 11 (6), 86–90.

Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M. and Thuraisingham, B.M., 2009a. A semantic web based framework for social network access control. *In*: B. Carminati and J. Joshi, eds. *14th ACM Symposium on Access Control Models and Technologies (SACMAT 2009)* ACM, 177–186.

Carminati, B., Ferrari, E., and Perego, A., 2009b. Enforcing access control in Web-based social networks. *ACM Trans. Inf. Syst. Secur.*, 13 (1), 1–38.

Carmo, J. and Jones, A., 2002. Deontic Logic and Contrary To Duties. *In*: D. Gabbay and F. Guenther, eds. *Handbook of Philosophical Logic, 2nd Edition.*, Vol. 8 Dordrecht: Kluwer, 265–343.

Governatori, G., 2004. Defeasible Description Logic. *In*: G. Antoniou and H. Boley, eds. *Third International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004)*, no. 3323 in Lecture Notes in Computer Science, Berlin, 98–112.

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

Governatori, G., Hall, J., and Paschke, A., eds., 2009. *International Symposium Rule Interchange and Applications (RuleML 2009)*, Vol. 5858 of *Lecture Notes in Computer Science* Springer, 2009.

Governatori, G., Hulstijn, J., Riveret, R. and Rotolo, A., 2007a. Characterising Deadlines in Temporal Modal Defeasible Logic. *In*: M. Orgun and J. Thornton, eds. *20th Australian Joint Conference on Artificial Intelligence, AI 2007*, Vol. 4830 of *Lecture Notes in Computer Science* Springer, 486–496.

Governatori, G. and Iannella, R., 2009. Modelling and Reasoning Languages for Social Networks Policies. *In*: *Enterprise Distributed Object Computing Conference (EDOC 2009)* IEEE, 193–200.

Governatori, G. and Milosevic, Z., 2006. A Formal Analysis of a Business Contract Language. *International Journal of Cooperative Information Systems*, 15 (4), 659–685.

Governatori, G., Milosevic, Z., and Sadiq, S., 2006. Compliance checking between business processes and business contracts. *In*: *10th International Enterprise Distributed Object Computing Conference (EDOC 2006)* IEEE Computing Society, 221–232.

Governatori, G. and Pham, D.H., 2009. DR-CONTRACT: an architecture for e-contracts in defeasible logic. *International Journal of Business Process Integration and Management*, 4 (3), 187–199.

Governatori, G. and Rotolo, A., 2006. Logic of Violations: A Gentzen System for Reasoning with Contrary-To-Duty Obligations. *Australasian Journal of Logic*, 4, 193–215.

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

Governatori, G. and Rotolo, A., 2010. On the Complexity of Temporal Defeasible Logic. *In*: *Non-Monotonic Reasoning 2010 (NMR 2010)*, CEUR Workshop Proceedings.

Governatori, G., Rotolo, A., Riveret, R., Palmirani, M. and Sartor, G., 2007b. Variants of Temporal Defeasible Logic for Modelling Norm Modifications. *In*: *11th International Conference on Artificial Intelligence and Law (ICAIL07)* ACM, 155–159.

Governatori, G., Rotolo, A., and Sartor, G., 2005. Temporalised Normative Positions in Defeasible Logic. *In*: *10th International Conference on Artificial Intelligence and Law (ICAIL05)* ACM, 25–34.

Governatori, G., Rotolo, A., and Rubino, R., 2010. Implementing Temporal Defeasible Logic for Modeling Legal Reasoning. *In*: *3rd Juris-Informatics Workshop (Jurisin 2009)*, LNAI Berlin: Springer.

Guth, S. and Iannella, R., 2009a. ODRL V2.0 — Common Vocabulary. ODRL Initiative. Working Draft, 25 September 2009. [online] `http://odrl.net/2.0/WD-ODRL-Vocab.html` [5 August 2010a].

Guth, S. and Iannella, R., 2009b. ODRL V2.0 — Core Model. ODRL Initiative. Draft Specification: 23 September 2009. [online] `http://odrl.net/2.0/DS-ODRL-Model.html` [5 August 2010b].

Iannella, R., 2002. Open Digital Rights Language (ODRL) Version 1.1. W3C Note 19 September 2002. [online] `http://www.w3.org/TR/odrl` [5 August 2010].

Iannella, R., 2009. Industry Challenges for Social and Professional Networks. *In*: *W3C Workshop on the Future of Social Networking*, Barcellona, 15–16 January. `http://www.w3.org/2008/09/msnws/papers/nicta-position-paper.pdf`.

Kolovski, V., Hendler, J., and Parsia, B., 2007. Analyzing web access control policies. *In*: C. Williamson, M. Zurko, P. Patel-Schneider and P. Shenoy, eds. *16th International Conference on World Wide Web (WWW 2007)* ACM, 677–686.

Kruk, S., Grzonkowski, S., Gzella, A., Woroniecki, T., and Choi, H.-C., 2006. D-FOAF: Distributed Identity Management with Access Rights Delegation. *In*: R. Mizoguchi, Z. Shi and F. Giunchiglia, eds. *First Asian Semantic Web Conference (ASWC 2006)*, Vol. 4185 of *Lecture Notes in Computer Science* Springer, 140–154.

Lam, H. and Governatori, G., 2009. The Making of SPINdle. Governatori *et al.* (2009) Springer, 315–322.

Lee, A., Boyer, J.P., Olson, L.E. and Gunter, C.A., 2006. Defeasible security policy composition for web services. *In*: *Fourth ACM workshop on Formal methods in security (FSME 06)* New York, NY, USA: ACM, 45–54.

Maher, M., 2001. Propositional Defeasible Logic has Linear Complexity. *Theory and Practice of Logic Programming*, 1, 691–711.

Mikroyannidis, A., 2007. Toward a Social Semantic Web. *IEEE Computer*, 40 (11), 113–115.

Ni, Q., Bertino, E., and Lobo, J., 2008. An obligation model bridging access control policies and privacy policies. *In*: I. Ray and N. Li, eds. *13th ACM Symposium on Access Control Models and Technologies (SACMAT 2008)* ACM, 133–142.

Nute, D., 1994. Handbook of Logic in Artificial Intelligence and Logic Programming. ., Vol. 3, Oxford, chap. Defeasible Logic, 353–395.

Passant, A., Kärger, P., Hausenblas, M., Olmedilla, D., Polleres, A. and Decker, S., 2009. Enabling Trust and Privacy on the Social Web. *In*: *W3C Workshop on the Future of Social Networking*, Barcellona, 15–16 January. `http://www.w3.org/2008/09/msnws/papers/trustprivacy.html`.

Rubino, R. and Rotolo, A., 2009. A Java Implementation of Temporal Defeasible Logic. Governatori *et al.* (2009) Springer, 297–304.

Sadiq, S., Governatori, G., and Naimiri, K., 2007. Modelling of Control Objectives for Business Process Compliance. *In*: G. Alonso, P. Dadam and M. Rosemann, eds. *5th International Conference on Business Process Management (BPM 2007)*, no. 4714 in LNCS Springer, 149–164.

Simpson, A., 2008. On the need for user-defined fine-grained access control policies for social networking applications. *In*: *Workshop on Security in Opportunistic and SOCial networks (SOSOC 08)* New York, NY, USA: ACM, 1–8.

Tonti, G., *et al.*, 2003. Semantic Web Languages for Policy Representation and Reasoning: A Comparison of Kaos, Rei, and Ponder. *In*: *2nd International Semantic Web Conference (ISWC2003)*, no. 2870 in Lecture Notes in Computer Science Springer, 419–437.