International Journal on Advanced Science Engineering Information Technology

Vol.7 (2017) No. 3
ISSN: 2088-5334

Defeasible Policy Language for Online Social Networks

Mahdi Rohaninezhad[#1], Shahrul Azman Mohd Noah[#2], Shereena Mohd Arif [#3]

[#] *Center of Artificial Intelligence Technologies, Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600, Bangi, Selangor, Malaysia*
E-mail: [1]rohaninezhad@gmail.com,[2]shahrul@ukm.edu.my, [3]shereen.ma@gmail.com

***Abstract*— Current online social network sites have not addressed policy control sufficiently. In addition, the existing rule-based proposals on policy control have not been able to cope with normative, temporal, exceptional and conflicting nature of OSNs policies. These characteristics of OSNs policies fit to defeasible logic formalism. Thus, we contextualized a defeasible policy language and proposed corresponding ontologies to extend an existing ontology framework on policy control called open digital right language. Our ontology proposal focused on OSNs use cases and provided solution for implementing norms, deadlines and conflict resolution and compensation models for policies. Deployed ontology shows that defeasible policy language is expressive enough to represent and manipulate complicated policy use cases of OSNs.**

***Keywords*— modal defeasible logic; policy language; online social networks.**

I. INTRODUCTION

Privacy is the main concern in online social networks (OSNs) [1-5]. The issue is deepening since various OSNs are appearing and each contains specific personal information of citizens where companies and organizations are eager to access and benefit. Thus, the policy control framework for usage and access control in OSN web sites and the underlying applications must be addressed. Although there have been a lot of researches on policy control for OSNs, the investigation on how to express and reason with conflicting, exceptional, normative and temporal policies has remained neglected. In this paper we contextualize a defeasible policy language (DPL) and propose corresponding ontology vocabularies to represent and reason with norms, deadlines and conflicts in OSNs policies.

Access control is the dominant model for conceptualizing and managing privacy policy in today's online world. Access control in OSNs mainly is regulated based on the degree of relationship, which in turn, stems from the notion of trust reflected in user-to-user relationship [6]. The Kruk et al. system, which is called D-FOAF, is the first ontology-based and FOAF-like model for OSNs that supports access control based on the trust concept. This relation-based interpretation and implementation of access control in OSNs became a cornerstone for managing privacy at the very beginning.

The proposal of Carminati et al. [7, 8] extended the previous relation-based approach by suggesting a rule-based model for access control based on type, depth and trust level of relationship in OSNs. They employed a rudimentary Horn-like language to regulate user's privacy policies. Carminati and colleagues enhanced their access control architecture towards semantic web tools in [7, 9]. Their main idea was to represent social network information through ontology language and then use reasoning tools of semantic web to manage privacy. In fact, they followed the Kruk proposal [6] and extended ontologies for; (1) user's profile, (2) resources (e.g., photo album), (3) actions, and finally (3) relationship beetween users and resources, (e.g., post a status). This provides the possibility of using semantic web reasoning tools such as SweetRules to do reasoning on access control. However, inference on permission can be complicated where the source of authorization is decentralized and abdicated to OSNs users. Thus, they had a naive discussion on conflict and exception resolution or express normative aspects of policies in their framework.

Governatori and Iannella [10] argued that existing ontological and rule-based proposals, which are based on first order language, fundamentally are not capable to address policy language. Because normative, exceptional and conflicting characteristic of policies are out of the boundaries of first order language. Thus, they suggested to extend ontological framework towards defeasible logic [11] formalism in [12]. They suggested defeasible logic to overcome policy conflicts and modal operators on defeasible logic to address normative and temporal aspects of policies. They also drew outline of extending an existing ontological framework on policy control called Open Digital Right Language (ODRL).

This work continued the outline to cover temporalised policies, policy conflict detection and compensation. We provided ontology vocabularies to infer and compensate policy conflicts as well as deadlines. Then we addressed some usage and access control policies of OSNs using suggested ontologies,. We employed ODRL ontology version 2.1 investigated in [13] which is available in [14] and [15]. In the following sections we first discussed about DPL components. Then we described ODRL ontology vocabularies and finally introduced proposed ontology vocabularies to map DPL components to ODRL.

II. Material and Method

In this section we contextualise DPL features based on Governatori and Iannella [12] proposal for policy control. We focus on specific OSNs policy use cases that are not addressed before. The use cases are brought from both access and usage control. Each subsection addresses one important gap of rule-based policy languages for OSNs.

A. Normative policy model

Policy languages are compliant with normative knowledge representation. And normative regulations can be represented under deontic logic [16] addresses concepts like obligation, permission and prohibition. These concepts correspond to notions such as duty, right and prohibition. Deontic logic extends first order logic to deontic operators O, P and F indicate obligation, permission and prohibition (forbidden) respectively. The following relations are satisfied with the above deontic operation,

$$OA \equiv \neg P \neg A \quad \neg O \neg A \equiv PA \quad O \neg A \equiv FA \quad \neg PA \equiv FA \quad (1)$$

The rule $OA \rightarrow PA$ is another implication that deontic operations satisfy. This relation indicates that if A is obligatory then it is permitted. In fact, we could ensure internal consistency of obligations in a set of norms, by examining the possibility of executing obligations without doing forbidden acts. Governatori [17] extended standard deontic logic towards directed deontic operations where subject and beneficiary of normative operators are specified.

Use case. "Tom forbids acquaintances to save photo"

In detail, $O_s^b A$ denotes an obligation where s represents the subject and b the beneficiary of normative operator. Thus, $O_s^b A$ means that s (e.g., a Facebook user) has the obligation of A (e.g., prevent access personal information) with respect to b (e.g., an acquaintances). Or $F_s^b B$ means that s (e.g, an acquaintance) is forbidden of B (e.g, usage private data) with respect to b (e.g., a Facebook user). Thus, the DPL provides a ground for implementing normative aspect of privacy policies. Indeed, deontic operations denote access/deny permission by obligation/prohibition operations.

B. Exception and conflict resolution model

Conflicts in OSNs policies stem from different facets of social life (i.e., the source of policies might vary) [18]. Superiority relation in standard defeasible logic is employed to capture conflicts and exceptions. Exception is a state that implements default notion where some additional information prevents to trigger normal policy condition. For instance, the following default policy,

Use case 1. "Facebook users are allowed to save other member photos, unless they are classified as private." This can be represented by the following default rule,

$$r1: photos(x) \Rightarrow P\ save(x) \quad (2)$$

The rule states that if x is a photo then the save permission is granted. Note that there is no need to represents users in the rule since it is implicitly covered. But to represent unless part of the policy condition we have,

$$r2: private(x)\ ,\ \neg owner(x,y) \Rightarrow O_y \neg save(x) \quad (3)$$

This indicates that if someone y is not owner of a photo x and the resource is a private then the usage of x is forbidden for y. Note that prohibition is expressed by obligation followed by negation here in this rule since prohibition is a negated obligation.

Thus, two rules are contradictory and no conclusion can be derived. Superiority relation represents and resolves the conflict in the logic. Having superiority relation $r1<r2$, the rule $r2$ defeats $r1$ when both rules are triggered.

This expressive power of defeasible logic can easily model conflicting policy conditions, which might come from different resources (OSNs or even offline life facets). The most privacy policies in OSNs are about specifying who can access which re-sources that can be classified as follows,

- Only the resource owner
- A friend
- A friend list
- The second level of friends (friends of friends)
- All friend lists
- Everybody

Thus, each policy language must be able to manage usage (access) control for all above cases. For example,

Use case 2. "Tome allowed all friend lists to save wedding photos except his colleagues". Then the usage control in this use case can be represented as follows,

$$r1: friend(y),\ wedding\_photo(x) \Rightarrow P_y save(x) \quad (4)$$

Expresses that if y is a friend can save wedding photo x. But the following rule for-bids colleagues to save the photo.

$$r2: colleague(y) \Rightarrow O_y \neg save(x) \quad (5)$$

Similar to the last example by employing superiority relation ($r1<r2$) the exceptional case can be managed.

C. Privacy violation resolution model

Although contrary-to-duty (CTD) obligation is a typical phenomenon in normative systems, of which privacy policies are particular instances, it is not addressed in OSNs yet. On the other hand, reasoning under CTD obligations is one of the main philosophical aspects of deontic reasoning.

Deontic logic of violations proposed in [19] addresses this particular part of deontic reasoning. A sequential order of CTD obligations is denoted by compensation operator $\otimes$ to be used like $OA \otimes OB$. It means that we have the obligation

of A (i.e., OA), but if this is violated, i.e., we have the negation of A, i.e., ¬A, then the obligation OB is enforced. Other words, obligation B is a compensate for obligation A if is failed to be fulfilled. In the same way a chain of obligation compensation can be formulated as follows,

$$OA_1 \otimes ... \otimes OA_n \quad (6)$$

The chain of obligation states that OA_1 is the main obligation, but if it is violated then the next obligation triggers and so on. It is worth noting that only obligations and prohibitions can appear in the chain. Permission can come only at the end of the chain. Because it is not possible to violate permission and compensation is only meaningful for statements that cannot be violated.

DPL supports the violation resolution model based on defeasible inference. In this language, the consequence of a defeasible rule can be a chain of obligation compensation. It is worth remembering that in standard defeasible logic, the consequence of a rule only can be a single literal not a chain of compensation. But DPL extends classical defeasible logic with compensation operator such that the obligation OA appears in conclusion. In the following defeasible rule of compensation obligation, to prove OA we have to prove negation of all elements appear before OA in the chain,

$$P_1,...,P_n \Rightarrow OB_1 \otimes ... \otimes OB_n \otimes OA \otimes OC_1 ... \otimes OC_n \quad (7)$$

All $P_1,...,P_n$ must be provable, and we must have $\neg B_1,...,\neg B_n$. For more details refer to [17].

Use case 3. If a user is reported as abuser then he must remove his offensive post, or inactive his account for six months otherwise will not have access permission to his account. The obligations can be represented as follows,

$$r1: abuser(x) \Rightarrow O_x remove(y) \otimes O_x inactive(x) \otimes P_x \neg access(x). \quad (8)$$

This indicates that if x is abuser, then he must remove his abusive post y or otherwise inactive his account for six months or otherwise will lose the permission to access his account. Technically, if the first condition, i.e., $O_x remove(y)$ is violated, the second condition must hold, i.e., $O_x inactive(x)$, otherwise user will lose his access to his account $P_x \neg access(x)$. This conclusion is contradictory with the general social network regulation that everyone has access to his account. This conflict can be resolved by using the exception and resolution model discussed in section *B*.

D. Temporal policy model

Time is one of the key factors in OSN policies since it is an important aspect of offline social life. For instance, a policy might be valid only for a year and compensate another policy for several months. Or an OSN website might interested to consider a deadline for an obligation in the site (e.g., if users are inactive for 6 months their account will be inactive automatically). It is essential to determine which policies might be in conflict in time axis. Temporalizing normative propositions first proposed by Governatori et al. in [20, 21]. They suggested RuleML like language for the temporal rules in [22] and [23] and showed the linear complexity of the logic in [24]. Then, Governatori et al. [20] represented the concept of deadline and enhanced it efficiently [12].

The main idea in this aspect of DPL is that each normative proposition is attached with a timestamp. Thus, in temporalizing DPL language each proposition p denotes by p^t that t is a timestamp. For example, $post(Tom, wedding\_pic, family\_list)^{20151013}$ means that Tom posted his wedding photo for family list on 13th of Oct 2015.

The second key specific of DPL is that conclusions can be either persistent or transient. Correspondingly, two classes of rules (persistent or transient) are conceivable. Persistent rules denoted as follows,

$$a_1^{t1}, a_2^{t1}, ..., a_n^{t1} \Rightarrow^{\pi} c^t \quad (9)$$

This implies that if a_1 hold on t_1 and a_2 hold on t_2 and so on for all elements of the rule in antecedent, then we can conclude c at t, and the conclusion is valid until it is terminated.

But the transient rules denoted as follows,

$$a_1^{t1}, a_2^{t1}, ..., a_n^{t1} \Rightarrow^{\tau} c^t \quad (10)$$

expresses the same semantic with the previous one except that the conclusion is valid only at the time t (not after that).

This rule's classification provides a ground to implement the concept of deadline in policy control. In fact, deadline is an obligation confined with temporal parameters in the context of policy languages. Deadlines also are classified into two distinguished types, namely, achievement obligation like and maintenance obligation like policies. Achievement obligation must happen at least once before the deadline (use case 4). But maintenance obligation, must hold during all time before deadline (use case 5).

Use case 4. Facebook users must complete their profile info after 30 days of opening account.

In this example, deadline refers to an obligation triggered by opening account ($open_{init}$), which is a persistent obligation. Then, user has to complete her profile before the deadline. The obligation automatically will be terminated only if the user obeys it ($open_{term}$). It is worth noting that obligation may remain persist even after the deadline (like this example). In this case, the role of deadline is to signal obligation violation ($open_{viol}$).

$open_{init}: open\_account(x)^{t1} \Rightarrow_{\pi} O_x complete\_profile(x)^{t1}$

$open_{term}: O_x complete\_profile(x)^{t2}, complete\_profile(x)^{t2} \Rightarrow_{\tau} \neg O_x complete\_profile(x)^{t2+1}$

$open_{viol}: open\_account(x)^{t1}, O_x complete\_profile(x)^{t1+30} \Rightarrow_{\tau} viol(open)^{t1+30}$

Having the fact set $\{open\_account(Tom)^1, complete\_profile(Tom)^{20}\}$ the rule $Open_{init}$ triggers the persistent obligation to complete profile. Then, having $complete\_profile(Tom)^{20}$ fact trigger the rule $open_{init}$ rule which terminates obligation in 21. Thus, the violation rule ($open_{viol}$) is not triggered on 30.

Use case 5. Facebook users must keep active in the last six month

In this Example, the deadline signals only when obligation terminates. And obligation terminates only if user keep active (logic in the last 6 months) to the web site ($keep_{term}$). If obligation does not occur in some times before the deadline, then violation occurs ($keep_{viol}$).

$keep_{init}$: $last\_login(x)^{t1} \Rightarrow_{\pi} O_x keep\_active(x)^{t1}$

$keep_{term}$: $O_x keep\_active(x)^{t1} \Rightarrow_{\tau} \neg O_x keep\_active(x)^{t1+180}$

$keep_{viol}$: $O_x keep\_active(x)^{t2}$, $\neg O_x keep\_active(x)^{t2} \Rightarrow_{\tau} viol(keep)^{t2}$

It is worth noting that there might be cases that maintenance obligations are undefined (i.e., not for 180 days) where no termination condition needed and the rule $keep_{term}$ will be dropped. Or might be cases that the termination condition exists but the exact time is not determined and depends on other events. For example, when termination condition is; if user insults other users in OSNs. In such a case we need to add another clause representing an event or action in the body of termination condition.

Maintenance obligations do not persist after the deadline, while often achievement obligations do until they achieved. Nevertheless, there might be the cases that achievement obligation also terminates after deadline. For more details, refer to [20].

III. Result and Discussion

A. ODRL Basic Components

In the past years, ODRL has extended from a digital right language to a wider range of policy language. Figure 1, depicts the core UML model of ODRL version 2.1. According to the Figure, the core model components include *policy*, *asset*, *party*, *permission*, *duty*, *prohibition*, *action* and *constraint*.

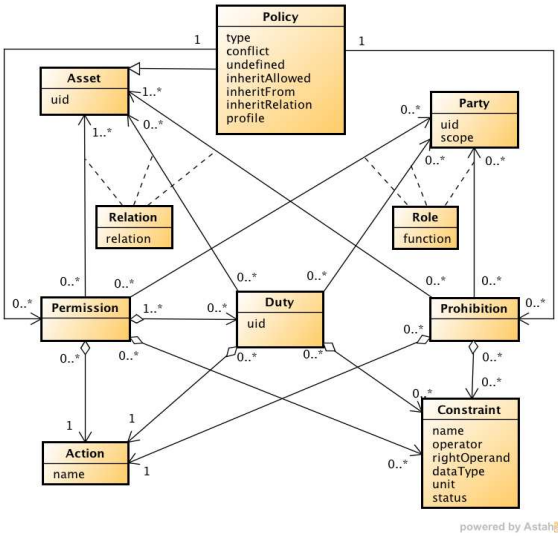

Fig 1. ODRL new version core model taken from ODRL community

Policy is the central top entity in the ODRL model, which refers to Permission and Prohibition that hold the Policy. This entity has several subclasses, namely, agreement, offer, privacy, set and ticket (For more detail see [14]). In the domain of social network, policies focus on who is the end Party; only the resource owner, a friend, a friend list, the second level of friends (friends of a friend), all friend lists or everybody.

Asset is the data object that its usage is confined by policy constraints. The assets in OSNs are user's resources like pictures, profile info and etc.

Party represents policy subjects that have two subtypes of Assigner (who establish policy constraints) and Assignee (who receive policy constraints). In OSNs they are users and their friends respectively.

Permission allows particular Action on a specific Asset. Permission is related to Parties, Constraints and probable Duties under which license granted. For example, share a picture in OSNs is to give permission of display to friends.

Prohibition is to forbid particular Action on a specific asset (opposite Permission).

Duty implies necessary Action that the Permission Assignee has to perform to have permission applicable.

Action is an operation that Assignee is allowed to do (if is about Permission), or forbidden to do (if is about Prohibition) or has to do (if is about Duty). Potential actions in OSNs are display, share, print or play of a picture (movie) and etc.

Constraints are limitations or conditions on Permissions, Prohibitions or Duties in the policy language. Conditions may contain *temporal*, *logical* or *mathematical* operators.

B. Mapping ODRL Ontology with DPL Language

Current ODRL ontology language[1] supports some basic elements of DPL including normative rules and conflicts resolution. As depicted in Figure 2, ODRL ontology provides a constructive model for deontic modalities (Permission, Prohibition and Obligation). In the following section we articulate rule, literals, predicates and terms, which are the basic vocabularies of DPL, based on ODRL ontology elements. Then we describe how ODRL ontology language can capture and implement other DPL specifics such as "conflict policy", "compensation policy" and "temporal policy".

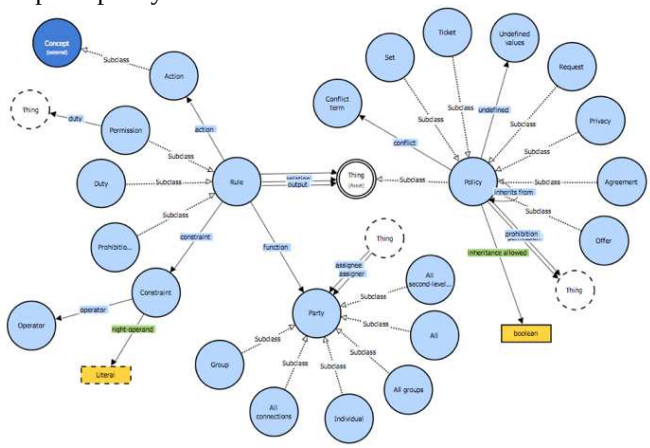

Fig 2. ODRL version 2.1 ontology visualized by VOWL.

Rule. Rule owl:Class is placed in the central position of ODRL policy language ontology that implements deontic modalities by *Permission*, *Prohibition* and *Duty* disjoined subclasses (see Fig. 2). This class can be mapped to deontic operators P, ¬O and O in DPL respectively. A Rule in

[1] The ODRL ontology URI is available in http://www.w3.org/ns/odrl/2/

ODRL indicates special Action that Assignee is permitted/prohibited (i.e., Permission/Prohibition) to do on an Asset under a specific Constraint or Duty. Thus, rule entity is the subject of Asset (i.e., Policy), Constraint (i.e., time), Party (i.e., Assignee and Assigner) and Action (e.g., download) entities. It is worth noting that each rule must include exactly one Action in ODRL language. And the range of deontic modal properties is the union of Action and the modal operator. Duty (Rule owl:subclass) applied only as a condition of Permission/Prohibition. Consequently, each head literal might bound by a deontic operator (i.e., Permission/Prohibition/Duty), Action, Assignee and Asset (i.e., `P Action(Assignee, Asset)`). And body literals may include Duty, Constraint, Asset and Assigner. The rule RDF vocabulary in ODRL is as follows,

```
<!--http://www.w3.org/ns/odrl/2/Rule-->
    <owl:Class rdf:about="Rule">
        <rdfs:label
xml:lang="en">Rule</rdfs:label>
        <rdfs:isDefinedBy rdf:resource=""/>
    </owl:Class>

<!--  http://www.w3.org/ns/odrl/2/Permission
-->
    <owl:Class rdf:about="Permission">
        <rdfs:label
xml:lang="en">Permission</rdfs:label>
        <rdfs:subClassOf
rdf:resource="Rule"/>
        <owl:disjointWith
rdf:resource="Prohibition"/>
        <rdfs:isDefinedBy rdf:resource=""/>
    </owl:Class>

<!--
http://www.w3.org/ns/odrl/2/Prohibition -->
    <owl:Class rdf:about="Prohibition">
        <rdfs:label
xml:lang="en">Prohibition</rdfs:label>
        <rdfs:subClassOf
rdf:resource="Rule"/>
        <rdfs:isDefinedBy rdf:resource=""/>
    </owl:Class>

<!--http://www.w3.org/ns/odrl/2/Duty-->
    <owl:Class rdf:about="Duty">
        <rdfs:label
xml:lang="en">Duty</rdfs:label>
        <rdfs:subClassOf
rdf:resource="Rule"/>
        <owl:disjointWith
rdf:resource="Permission"/>
        <owl:disjointWith
rdf:resource="Prohibition"/>
        <rdfs:isDefinedBy rdf:resource=""/>
    </owl:Class>
```

Policy conflict resolution. In ODRL, policy conflict arises when contradictory Actions appear in Permission/Prohibition. The instances of "Conflict term" owl:Class (i.e, Perm, Prohibit and Invalid) characterize the policies to resolve conflicts in ODRL version 2.1. Perm indicates that Permission is considered precedence, but Prohibit gives priority to Prohibition and Invalid indicates that the license is not applicable (this called precedent mechanism). Thus, the notion of conflict resolution of DPL language can easily be implemented by ODRL ontology framework. However, the "Conflict term" implementation in ODRL is not able to capture some notions behind superiority relation in DPL since its vocabulary does not support rule labeling. In fact, it is not going to establish superiorities between different policies but only consider priorities within a policy (risen by Prohibition/Permission). The conflict term vocabulary in ODRL is as follows:

```
<!--http://www.w3.org/ns/odrl/2/conflict-->
    <owl:ObjectProperty
    rdf:about="conflict">
      <rdfs:label
    xml:lang="en">conflict</rdfs:label>
      <rdfs:range
    rdf:resource="ConflictTerm"/>
      <rdfs:domain rdf:resource="Policy"/>
</owl:ObjectProperty>
```

Policy conflict deduction. One of the main challenges in ODRL ontology making is to create ontology vocabulary such that is able to detect (infer) conflicts in Permission/Prohibition. On the one hand, a friend (Assignee) might place in several friend lists where each one may involve in different usage (or access) policy. On the other hand, a resource (Asset) might be subject of different conflicting Policies come from different resources. Thus one source of conflict is Assignee and the other one is Asset. For example, one policy might state that all Alice close friends can access to all photos while another policy expresses that only close friends who are invited to her wedding are allowed to access her wedding photos. To detect conflicts in OSNs we suggest changing Party class in ODRL. We first create different friend lists as a subclass for Group (e.g., Friend, Colleague, Family and etc). Then we need to employ Individual as a class restriction for friend lists. By doing this, Friend is a class of instances of Individuals who participate in friend property (using anonymous defined class). Thus, each Group Policy assignment is reducible to Individual Policy upon which conflicting policies can be inferred and noticed to user. For example, Family RDF vocabulary will be as follows:

```
<!-- http://www.w3.org/ns/odrl/2/Family -->
    <owl:Class rdf:about="Family">
        <rdfs:subClassOf
rdf:resource="Group"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="isFamily"/>
                <owl:allValuesFrom
rdf:resource="Individual"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
```

This expresses that Family are individuals that has isFamiliy property. Thus, the result of querying use case 2 is as follows:

```
  Individual:
http://www.w3.org/ns/odrl/2/Policy:02
      Types:
          Agreement
      Facts:
          prohibition
http://www.w3.org/ns/odrl/2/Prohibit:02,
          conflict  Prohibition,
          permission
http://www.w3.org/ns/odrl/2/Perm:02


  Individual:
http://www.w3.org/ns/odrl/2/Perm:02
      Types:
          Permission
      Facts:
          assigner
http://www.w3.org/ns/odrl/2/user:02,
          action  save,
          target
http://www.w3.org/ns/odrl/2/weddingPhoto:02


  Individual:
http://www.w3.org/ns/odrl/2/Prohibit:02
      Types:
          Prohibition
      Facts:
          assigner
http://www.w3.org/ns/odrl/2/user:02,
          action  save,
        target
http://www.w3.org/ns/odrl/2/weddingPhoto:02
```

This policy includes two conflicting Permission and Prohibition policies at which Prohibition has higher priority. This indicates that if a friend is the subject of both policies then he could not save the wedding photos.

But we should notice that in the above example Assignees are not determined. Indeed, Assignees are all Colleague and Friend instances that are determined through hasValue restriction as follows:

```
<!-- http://www.w3.org/ns/odrl/2/Friend -->
      <owl:Class rdf:about="Friend">
          <rdfs:subClassOf
rdf:resource="Group"/>
          <rdfs:subClassOf>
              <owl:Restriction>
                  <owl:onProperty
rdf:resource="assignee"/>
                  <owl:hasValue
rdf:resource="Perm:02"/>
              </owl:Restriction>
          </rdfs:subClassOf>
          <rdfs:subClassOf>
              <owl:Restriction>
                  <owl:onProperty
rdf:resource="isFriend"/>
                  <owl:allValuesFrom
rdf:resource="Individual"/>
              </owl:Restriction>
          </rdfs:subClassOf>
      </owl:Class>

<!--   http://www.w3.org/ns/odrl/2/Colleague
-->
      <owl:Class rdf:about="Colleague">
          <rdfs:subClassOf
rdf:resource="Group"/>
          <rdfs:subClassOf>
              <owl:Restriction>
                  <owl:onProperty
rdf:resource="assignee"/>
                  <owl:hasValue
rdf:resource="Prohibit:02"/>
              </owl:Restriction>
          </rdfs:subClassOf>
          <rdfs:subClassOf>
              <owl:Restriction>
                  <owl:onProperty
rdf:resource="isColleague"/>
                  <owl:allValuesFrom
rdf:resource="Individual"/>
              </owl:Restriction>
          </rdfs:subClassOf>
      </owl:Class>
```

This expresses that Friends are instances of Individual class who can save wedding photos whereas Colleagues cannot. Thus, we easily can query all individual's policy and detect and notify probable conflicts in between.

Policy compensation. As discussed in the section *C*, Duty compensation policy is denoted by compensation operation $\otimes$ in DPL. However, we can deploy compensation without this operator in ODRL. Because the rule $A \Rightarrow OB \otimes OC$ corresponds these two rules; $A \Rightarrow OB$, $A, \neg B \Rightarrow OC$. In ODRL ontology, Duty is assumed to be compulsory and Duty violation is not tolerated. In detail, if Assignee complies Duty then he will have the Permission to access to the Asset otherwise cannot. We suggest extending ODRL model to implement precedent attribute. This attribute establishes connection between individuals of Duty class. isPrecedent property is defined irreflexive to prevent meaningless Duty chain. The sequence order of isPrecedent conveys priorities between different Duties. Policy compensation vocabulary is provided in the following:

```
<!-- http://www.w3.org/ns/odrl/2/isPrecedent
-->
      <owl:ObjectProperty
rdf:about="isPrecedent">
          <rdf:type
rdf:resource="&owl;IrreflexiveProperty"/>
          <rdfs:range rdf:resource="Duty"/>
          <rdfs:domain rdf:resource="Duty"/>
      </owl:ObjectProperty>
```

For example, ontological representation of the use case 4 would be as follows:

```
  Individual:
http://www.w3.org/ns/odrl/2/Policy:04
      Types:
          Agreement
      Facts:
          permission
http://www.w3.org/ns/odrl/2/Perm:04

  Individual:
http://www.w3.org/ns/odrl/2/Perm:04
      Types:
          Permission
      Facts:
          assigner  Facebook,
          action  Access,
          target
http://www.w3.org/ns/odrl/2/account:04,
          duty
http://www.w3.org/ns/odrl/2/requirements:04,
          assignee
http://www.w3.org/ns/odrl/2/user:04


  Individual:
http://www.w3.org/ns/odrl/2/requirements:04
      Types:
          Duty
      Facts:
          isPrecedent
http://www.w3.org/ns/odrl/2/requirements:14,
          action  remove

  Individual:
http://www.w3.org/ns/odrl/2/requirements:14
      Types:
          Duty
      Facts:
        action  inactive
```

This expresses that the Duty instance "`requirements:04`" is related to the other one "`requirements:04`" trough precedent attribute where it is assumed that the former has higher priority but is relaxable by the later Duty. Thus, during applying policy we simply can query and check chain of Duties to implement compensation notion.

Temporal Policy. ODRL ontology vocabulary supports temporal Constraints using dateTime, time and right operators by which Permission/Prohibition can determine deadline. Other words, Constraint (owl:class) can limit Permission/Prohibition policy using "dateTime". Time interval also has implemented by employing right operators called gteq and lteq in ODRL. But the discussed deadline in ODRL only concerns Permission/Prohibition (e.g., "can" have access in deadline) not obligation or Duty (e.g., "must" complete the profile in deadline) whereas DPL language characterizes time obligations for Duties as well. In detail, Duty in ODRL ontology plays the role of requirement for Permission and duty attribute only appears in relation with Permission, whereas Duty in deadline is not going to grant/deny permission. Duty in here is to implement time obligation on an Action. Thus, Duty is triggered by initial action (init) and then obliges another action within a deadline (terms and conditions) otherwise obligation will be violated. Thus, we defined another type of Policy called Deadline accompanied with deadline attribute having three sub-properties called init, term and viol. Deadline attribute connect Policy to union of Duty and Action classes. Since every action attached with timestamp in DPL deadline use cases we suggest considering dateTime data property for Action in ODRL. We also considered "relax" data attribute for Duties to denote Duties that are done within the deadline. The suggested deadline vocabulary is as follows:

```
<!--http://www.w3.org/ns/odrl/2/deadline  -
->
    <owl:ObjectProperty
rdf:about="deadline">
<rdfs:domain rdf:resource="Policy"/>
        <rdfs:range>
            <owl:Class>
                <owl:unionOf
rdf:parseType="Collection">
                    <rdf:Description
rdf:about="Action"/>
                    <rdf:Description
rdf:about="Duty"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:range>
    </owl:ObjectProperty>

<!-- http://www.w3.org/ns/odrl/2/term -->
    <owl:ObjectProperty rdf:about="term">
        <rdfs:subPropertyOf
rdf:resource="deadline"/>
    </owl:ObjectProperty>

<!-- http://www.w3.org/ns/odrl/2/viol -->
    <owl:ObjectProperty rdf:about="viol">
        <rdfs:subPropertyOf
rdf:resource="deadline"/>
    </owl:ObjectProperty>

<!-- http://www.w3.org/ns/odrl/2/init -->
    <owl:ObjectProperty rdf:about="init">
        <rdfs:subPropertyOf
rdf:resource="deadline"/>
    </owl:ObjectProperty>
```

Thus, the use case 4 can be represented by ODRL ontology as follows:

```
<http://example.com/policy:07>
        odrl:viol
<http://example.com/duty:27> ;
        odrl:term
<http://example.com/duty:17> ;
        odrl:init
```

```
<http://example.com/duty:07> ;
        rdf:type   odrl:Deadline ;
        rdf:type   owl:NamedIndividual .
<http://example.com/duty:07>
        odrl:action    odrl:Open ;
        odrl:assigner  odrl:Facebook ;
        odrl:assignee
<http://example.com/user:07> ;
        odrl:action
<http://example.com/account:07> ;
        odrl:dateTime            "2016-04-
05"^^xsd:date ;
        rdf:type       odrl:Duty ;
        rdf:type
owl:NamedIndividual .

<http://example.com/duty:17>
        odrl:action      odrl:profile ;
        odrl:action      odrl:Complete ;
        odrl:constraint
<http://example.com/constraint:17> ;
        odrl:relax       true ;
        odrl:dateTime            "2016-04-
28"^^xsd:date ;
        rdf:type         odrl:Duty ;
        rdf:type
owl:NamedIndividual .
<http://example.com/duty:27>
        odrl:action  odrl:Close ;
        odrl:action
<http://example.com/account:07> ;
        odrl:relax   true ;
        rdf:type     odrl:Duty ;
        rdf:type     owl:NamedIndividual .
```

Where `policy:07` expresses that `user:07` has to complete his profile within 30 days from opening day (`2016-04-05`). Since user has completed the profile at `2016-04-28` the relax field of the “term” and “viol” is Duty set to true which indicates they are not remaining duties.

Unlike Permission/Prohibition policies that are set up once, Deadline policy fields might be completed in several steps. For instance, system may schedule to query Deadline policies every day to look for Duty terms that still are not relaxed and their due date is reached to execute and fill viol Duty fields. Or when the user completes his profile “relax” fields in term and viol should be updated. Proposed ontology works for both persistent and transient types of temporal policies.

IV. Conclusions

In this work, we extended ODRL ontology vocabularies towards DPL by which addressed normative, temporal, exceptional and conflicting policies based on modal defeasible logic formalism. This empowered us to represent complicated policy use cases on OSNs such that able to address privacy concerns and policy management in a broader view. Policy conflict resolution and compensation models as well as deadlines were explored by querying use cases in the framework. Indeed, we can query policy instances using SPARQL and parse them into modal defeasible logic to infer and pull out implicit indications of provided policies.

It is worth mentioning that an only part of policy implications is obtainable using ontology vocabularies and OWL description logic. Other aspect of reasoning with normative, temporal and conflicting policies is to define defeasible rules profile and superiority relations then apply them into ontological knowledge bases. This may happen using defeasible logic reasoning engines that support semantic web technology such as DR-Device and SPINdle. This scheme is quite feasible for heterogeneous OSNs since time complexity of defeasible logic is linear and it is employed in dealing with big data under semantic web technology. The suggested scheme is the subject of our future work.

Acknowledgment

We would like to thank CAIT research group for facilitating this project by offering lab and server. This work is funded under grant LRGS/TD/2011/UITM/ICT/01/02 and FRGS/2/2013/ICT02/UKM/02/2 by National University of Malaysia.

References

[1] B. Debatin, J. P. Lovejoy, A.-K. Horn, and B. N. Hughes, "Facebook and online privacy: Attitudes, behaviors, and unintended consequences," Journal of Computer,ÄêMediated Communication, vol. 15, pp. 83-108, 2009.

[2] N. B. Ellison, J. Vitak, C. Steinfield, R. Gray, and C. Lampe, "Negotiating privacy concerns and social capital needs in a social media environment," in Privacy Online, ed: Springer, 2011, pp. 19-32.

[3] J. V. E. Peluchette, K. Karl, and J. Fertig, "A Facebook ‘friend’ request from the boss: Too close for comfort?," Business horizons, vol. 56, pp. 291-300, 2013.

[4] F. Stutzman, J. Vitak, N. B. Ellison, R. Gray, and C. Lampe, "Privacy in Interaction: Exploring Disclosure and Social Capital in Facebook," in International Conference on Weblogs and Social Media (ICWSM'12), Dublin, IE, 2012.

[5] J. Vitak, C. Lampe, R. Gray, and N. B. Ellison, ""Why won't you be my Facebook friend?": strategies for managing context collapse in the workplace," presented at the Proceedings of the 2012 iConference, Toronto, Ontario, Canada, 2012.

[6] S. R. Kruk, S. Grzonkowski, A. Gzella, T. Woroniecki, and H.-C. Choi, "D-FOAF: Distributed identity management with access rights delegation," in The Semantic Web–ASWC 2006, ed: Springer, 2006, pp. 140-154.

[7] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "A semantic web based framework for social network access control," in Proceedings of the 14th ACM symposium on Access control models and technologies, 2009, pp. 177-186.

[8] B. Carminati, E. Ferrari, and A. Perego, "Rule-based access control for social networks," in On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, 2006, pp. 1734-1744.

[9] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Semantic web-based social network access control," computers & security, vol. 30, pp. 108-115, 2011.

[10] G. Governatori and R. Iannella, "Modelling and reasoning languages for social networks policies," in Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International, 2009, pp. 193-200.

[11] D. Nute, "Defeasible logic," in Handbook of logic in artificial intelligence and logic programming (vol. 3), ed: Oxford University Press, Inc., 1994, pp. 353-395.

[12] G. Governatori and R. Iannella, "A modelling and reasoning framework for social networks policies," Enterprise Information Systems, vol. 5, pp. 145-167, 2011.

[13] S. Steyskal and A. Polleres, "Defining expressive access policies for linked data using the ODRL ontology 2.0," in Proceedings of the 10th International Conference on Semantic Systems, 2014, pp. 20-23.

[14] R. M. Iannella, McRoberts; Víctor, Rodríguez Doncel. (2015, 27 Oct). ODRL Version 2.1 Ontology.
[15] S. Steyskal and A. Polleres, "Towards Formal Semantics for ODRL Policies," in Rule Technologies: Foundations, Tools, and Applications, ed: Springer, 2015, pp. 360-375.
[16] G. H. Von Wright, "Deontic logic," Mind, pp. 1-15, 1951.
[17] G. Governatori, "Representing business contracts in RuleML," International Journal of Cooperative Information Systems, vol. 14, pp. 181-216, 2005.
[18] M. Rohaninezhad, S. M. Arif, and S. A. M. Noah, "Defeasible Logic-Based Strategies to Regulate Facebook," Journal of Applied Sciences, vol. 14, pp. 2953-2966, 2014.
[19] G. Governatori and A. Rotolo, "Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations," 2005.
[20] G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo, "Characterising deadlines in temporal modal defeasible logic," in AI 2007: Advances in Artificial Intelligence, ed: Springer, 2007, pp. 486-496.
[21] G. Governatori and P. Terenziani, "Temporal extensions to defeasible logic," in AI 2007: Advances in Artificial Intelligence, ed: Springer, 2007, pp. 476-485.
[22] M. Palmirani, G. Governatori, and G. Contissa, "Temporal Dimensions in Rules Modelling," in JURIX, 2010, pp. 159-162.
[23] M. Palmirani, G. Governatori, and G. Contissa, "Modelling temporal legal rules," in Proceedings of the 13th International Conference on Artificial Intelligence and Law, 2011, pp. 131-135.
[24] G. Governatori and A. Rotolo, "On the complexity of temporal defeasible logic," in 13 International Workshop on Non-Monotonic Reasoning (NMR 2010), 2010.