Operational Semantics of an Extension of ODRL Able to Express Obligations

Nicoletta Fornara[1](✉) and Marco Colombetti[2]

[1] Università della Svizzera italiana, via G. Buffi 13, 6900 Lugano, Switzerland
nicoletta.fornara@usi.ch

[2] Politecnico di Milano, piazza Leonardo Da Vinci 32, Milan, Italy
marco.colombetti@polimi.it

Abstract. Nowadays economy is every day more and more a digital economy where many human activities are performed by means of digital devices. Those digital activities produce and operate on a big amount of digital assets, as the data stored in datasets, documents, images, videos or audio files. Rationally, it is useless that digital assets are made public without the specification of constrains on their usage and access. Many formal languages for expressing licenses, policies, norms, agreements, and contracts have been proposed in literature. Among them, the Open Digital Rights Language (ODRL) is a quite general one. In this paper, we present an extension of the syntax of ODRL for expressing conditional obligations. We present also an operational semantics of this extension with the goal of being able to perform automatic reasoning on the dynamic evolution in time of obligations. The definition of such operational semantics will be based on the specification of the lifecycle of obligations and on the definition of the mechanisms for computing their state using automatic reasoning. In particular, for doing that we use as far as possible, W3C standards: RDF and RDF Schema for the specification of obligations, and the Apache Jena general purpose rule engine for efficiently deducing the state of obligations on the bases of the state of the interaction among agents.

1 Introduction

Nowadays economy is every day more and more a digital economy where many human activities are performed by means of digital devices. Those digital activities produce and operate on a big amount of digital resources, as structured data stored in various datasets, documents, images, videos or audio files. Those digital resources may become digital assets when they are exchanged between peer agents or made available to data consumers, for example on the Internet, by data publishers.

Rationally, it is useless that digital assets are made public without the specification of constrains on their usage and access, this because the absence of a

© Springer Nature Switzerland AG 2018
F. Belardinelli and E. Argente (Eds.): EUMAS 2017/AT 2017, LNAI 10767, pp. 172–186, 2018.
https://doi.org/10.1007/978-3-030-01713-2_13

license or of a document attached to the data is not equivalent to the right to do whatever one wants with those data [11]. Nowadays, digital assets may be made available as Open Data (by adopting one of the Open Data Commons (ODC) licenses), or they can be associated to one of the Creative Commons licenses, or to other existing licenses like for example the Open Government License (OGL). Many other licenses with big or slight differences are created every day; therefore a standardization of the existing licenses with the goal of limiting their number to a finite set is far from the present situation [9]. Moreover, it is important to consider that instead of using pre-defined data licenses, data producers or data publishers may exchange digital assets with data consumers on the basis of an ad hoc agreement that may be reached by the parties in different ways, for example through a negotiation, by accepting an offer, by buying a ticket, and so on.

Licenses and agreements may be specified using human-readable formats, but the increasing usage and exchange of digital assets requires more formal and machine-readable mechanisms for the specification of licenses and agreements. This in order to enable machine-to-machine interactions combined with a number of useful services. Services like for example: (i) an advance search of resources based on their license; (ii) the possibility to aggregate different resources released under different licenses by computing license compatibility or conflicts; (iii) the automatic checking of the satisfaction or violations of the normative or legal relations that such an exchange of digital assets creates in the chain of interactions among data producers, data publishers, and data consumers.

In order to perform many of these services it is crucial not only to propose the syntax of a formal language for expressing licenses, policies, norms, and agreements, but also to express a formal semantics of such a language. This with the goal of being able to perform automatic reasoning on the normative or legal constrains, and to monitor the fulfilment or violation of a set of policies on the basis of the (usually partial) knowledge of the actions of the parties and of the state of their interaction. A formal semantics may be used also to simulate what will happen if one of the parties, related by a set of policies, perform certain actions.

Formal languages for expressing licenses (that can be used to waive some rights on a given resource), norms, agreements, and contracts have been developed in the Multiagent Systems (MAS) research community, and in its Normative MAS sub-community. Studies on Access Control Policies have been developed by the Access Control community; and studies on deontic logic developed by the Artificial Intelligence and Law community. Numerous of these proposals, which are related to this paper, are discussed in the Related Work section.

Among such formal languages, the most interesting and general one is the Open Digital Rights Language (ODRL), which uses Semantic Web languages for the formalization of policies. In its most recent version ODRL, as a language for expressing policies, presents some limits in its expressivity. In order to overcome one of these limits, in this paper we present an extension of the syntax of ODRL for expressing conditional obligations. We present also an operational semantics of this extension with the goal of being able to perform automatic reasoning on the temporal evolution of obligations. The definition of such operational semantics will be based on the specification of the lifecycle of obligations and on the definition of the mechanisms for computing the state of obligations using automatic reasoning. In particular, for doing that we use, as far as possible, W3C or de facto standards for the Semantic Web: RDF and RDF Schema for the specification of obligations, and the Apache Jena general purpose rule engine for efficiently deducing the state of obligations on the bases of the state of the interaction among agents.

The paper is organized as follows. In Sect. 2 formal languages for expressing licenses and agreements or contracts are presented, with a particular focus to the approaches that use semantic web technologies. In Sect. 3 we present the extension of the syntax of ODRL for expressing conditional obligations, and in Sect. 4 an operational semantics of such an extension is introduced. In Sect. 5 the implementation of a system able to simulate the evolution in time of a set of conditional obligations is evaluated and some conclusions and proposals for future work are discussed.

2 Related Work

Formal languages for expressing licenses (that can be used to waive some rights on a given resource) and agreements or contracts have been developed. Examples of such languages are: the MPEG21-Right Expression Language, which is mostly used for expressing rights on audio and video files; ccREL [1], an RDF-based language for expressing Creative Common licenses; and the Web Access Control (WAC) vocabulary[1], which provides a basic way to describe various forms of access to resources for users or groups who are identified by HTTP URIs.

Among such languages, the Open Digital Rights Language (ODRL) is the most general one, in that it is not focused on a file type nor only on access control. It can be used in different scenarios, as proved by the specification using ODRL 2.0 of the 126 licenses stored in the RDFLicense dataset [17]. Originally (in 2001), ODRL was an XML language for expressing digital rights, that is, digital content usage terms and conditions. In 2012, with version 2.0, and in 2015, with version 2.1 [10], ODRL evolved into a more general policy language: it is no longer focused only on the formalization of rights expressions, but also on the specification of privacy statements, like for example duties, permissions, and prohibitions. In version 2.0 and 2.1 ORDL is a Policy Language formalized in RDF with an abstract model specified by an ontology, and is developed and promoted by the ODRL W3C Community Group. In March 2016 the W3C "Permissions and Obligations Expression" (POE) Working Group was created with the goal of bringing the Community Group specifications through the W3C Process to "Recommendation" status. ODRL 2.1 has an informal semantics described in English in the Core Model, but there is not a formal specification of its semantics [19]. In [13] an OWL representation of ODRL 1.1 is presented, but it is

[1] http://www.w3.org/wiki/WebAccessControl.

limited to the representation of classes and properties, and no representation is given of the dynamic semantics of policies, that is, of how they evolve in time.

L4LOD (Licenses for Linked Open Data) is a lightweight vocabulary for expressing the licenses for Linked Open Data. It is presented in [9]. This language is able to express the semantics of the deontic component of licenses by using an extension of Defeasible Logic [8]. This extension is a non-monotonic logic able to take into account the idea of expressing obligations as defeasible rules, prohibitions as obligations to not do an action, and a simple notion of permission defined as defeaters of obligations (i.e. defeaters of prohibitions). The goal of the proposed work is to automatically compute the deontic components of a composite license l_c that can be obtained from a set of licenses $l_1,\ldots,l_n$. This is done by using deontic rules (having different type) and two heuristics for OR-composition and AND-composition of the deontic effects of licenses. The content of the formalized licenses is expressed using propositions like ShareAlike or Attribution, or Commercial. This is an interesting approach based on a non-monotonic logic that can be used only when it is not necessary to explicitly express the actor, the context, and other relevant attributes of the regulated actions.

In the field of studies on Access Control to data, an important XML standard language is the eXtensible Access Control Markup Language XACML [15] which is focused on the specification of access control policies. It is an important application independent policy language which is an OASIS industry standard. It is primarily an Attribute-Based Access Control system and an XML-based language for expressing and interchanging access control policies over the Web. It describes both a policy language and an access control decision request/response language. Policies in XACML are characterized by subjects, resources, environments, and actions. A subject element is the entity requesting access and it has one or more attributes. The resource element is a data, service, or system component and it has one or more attributes. The action element defines, by means of one or more attributes, the type of access requested on the resource. An environment element can optionally provide additional information. The formalization of access control obligations is barely supported by the XACML standard; indeed XACML only describes a general syntax for obligation specification, but focusses its core functionalities on permissions and prohibitions.

Among all these rights, policies, and access control languages, ODRL 2.1 is the most general one; in fact, it allows the specification of licenses, but also of more customizable offers of digital assets by data-provider, requests of digital data by data-consumer, or agreements that binds two parties.

The studies on Normative Multiagent Systems (NorMAS), concern mainly the formalization of norms used for expressing obligations, permissions, and prohibitions, and the definition and the realization of fundamental functionalities for norms promulgation, monitoring, enforcement, and for norms adoption and reasoning. In NorMAS literature there are various proposals for the specification of norms and policies using different languages [4,5,18] and of frameworks [3] for their management. In this paper, we propose to formalize policies using Semantic Web Technologies and therefore here we will mainly discuss other approaches where Semantic Web languages were adopted.

In [5,6] a proposal to specify and reason on commitments and obligations using OWL 2, SWRL rules, and OWL-API is presented (as widely discussed in next sections). In particular, in those papers an OWL ontology of obligations whose content is a class of possible actions that have to be performed within a given deadline is presented. The monitoring of those obligations (checking if they are fulfilled of violated on the basis of the actions of the agents) can be realized thanks to a specific framework required for managing the elapsing of time and to perform closed-world reasoning on certain classes. Unfortunately, the scalability of this approach is not good enough to make it usable in real applications.

Another interesting approach that uses Semantic Web Technologies for norms formalization and management is the OWL-POLAR framework for semantic policy representation and reasoning [18]. This framework investigates the possibility of using OWL ontologies for representing the state of the interaction among agents and SPARQL queries for reasoning on policies activation, for anticipating possible conflicts among policies, and for conflicts avoidance and resolution. In the OWL-POLAR model the activation condition and the content of the policies (that is what is prohibited, permitted or obliged by the policy) are represented using conjunctive semantic formulas, that is a conjunction of atomic assertions expressed using the concepts and the relations defined on an OWL ontology. Reasoning on a set of policies for deducing their state is quite expensive in OWL-POLAR, because it requires translating the activation condition and the content of a policy into the SPARQL query language and then evaluating the resulting queries on the OWL ontology used for representing the state of the world. In OWL-POLAR, there is no treatment of time. An interesting contribution of this framework is the study of conflicts among norms and a proposal for conflict resolution. The reasoning mechanisms proposed in this work are decidable, but if OWL DL is used, they are not tractable, in the worst case. If expressiveness is reduced to OWL Lite, decidability is guaranteed.

Another relevant proposal, where Semantic Web technologies are used for policy specification and management, is the KAoS policy management framework [3,20]. It is composed by three layers: (i) the human interface layer where policies, expressing authorizations and obligations, are specified in the form of constrained English sentences; (ii) the policy management layer used for encoding in OWL the policy-related information; (iii) the policy monitoring and enforcing layer used to compile OWL policies to an efficient format usable for monitoring and enforcement. Another approach where Semantic Web technologies are partially used is the Rei policy language [12] for modelling the deontic concepts of rights, prohibitions, obligations and dispensations. Rei is implemented in the logic language Prolog, but also includes some ontologies that enable the policy engine to interpret a subset of RDFS policies.

It is therefore clear that, even if the concept of norms and policies has been studied quite extensively in Multiagent research, a lot of work has to be done for improving the existing models and languages for norm and policy formalization. It is necessary to evaluate their expressivity on real-world use cases, to improve the performances and the scalability of the framework developed for policies management, i.e., for their creation, enforcement, and monitoring, and for making those approaches usable by practitioners and industry. This paper goes in this direction.

3 A Model of Obligation

Among the formal languages for expressing licenses, or more general normative or legal relations among autonomous parties/agents, presented in the previous section, ODRL is one of the most interesting, because of the following crucial characteristics: it is under study for becoming a W3C Recommendation, it has the possibility to be extended with profiles, there is an RDF specification of the Core Model of ODRL 2.1, and it has useful policy types like agreement, offer, request and ticket, that allow to manage personalized form of contracts for the distribution of digital assets. The last aspect is essential in all frequent situations where standard licenses do not meet the constraints that a data producer or a data provider may want to express.

In the ODRL 2.1 Core Model[2] a policy must contain at least one *permission* and may contain *prohibitions*. A permission may be conditioned by a *duty*. Permissions, prohibitions, and duties relate two *parties*, an *asset*, and an *action*. ODRL 2.1 has an informal semantics described in English in the Core Model. As a language for expressing licenses and more complex agreements or contracts, ODRL presents some limits in its expressivity, as it is clarified below and has been initially discussed in [2], where some syntactic changes to ODRL 2.0 were proposed.

In this section, we present a proposal to extend the syntax of ODRL in order to be able to express pure *obligations* to perform a certain class of actions. This is an important extension of ODRL 2.1 where it is only possible to express the duty to perform a specific action, as a requirement that must be fulfilled for obtaining a valid permission to perform another action.

Obligations are crucial for expressing the normative relationships in which an agent will become obliged to do a certain type of action a_1 as a consequence of the performance of another action a_2, where a_2 may be viewed as the *activation condition* of the obligation. Two concrete examples of this type of obligations are the following. A nurse may have the *possibility* to use an “emergency account” for getting access to sensitive health data of a patient, in case the emergency account is used the nurse becomes obligated to write within one week a report explaining the reasons why the account has been used. Another interesting example is given by the policy that regulates the access to the “limited traffic area” of certain big cities: people have the *possibility* to enter the “limited traffic area”, but if they do so they become obligated to pay a certain sum within a given temporal limit, otherwise they will be sanctioned. Obligations without activation conditions are rare and it is quite hard to find interesting examples.

[2] https://www.w3.org/community/odrl/model/2.1/.

The temporal relationship between the activation condition and the consequent obligation makes it impossible to formalize such obligations using only the notion of permission or prohibition. For example, a conditional permission stating that if the nurse writes the report she gets the permission to use the emergency account, would change the temporal relation among the actions, and therefore is not equivalent to the conditional obligation described above. Similarly, if a prohibition coupled with a sanction is used (stating that using the emergency account is prohibited, and in case the prohibition is violated, the sanction of writing a report applies) the result is a counterintuitive specification. Moreover, the formalization by means of a prohibition will create a violation that may affect the reputation of the violator, even if this is not a desired consequence.

In the ODRL 2.1 core model, it is assumed that "any use not explicitly permitted is prohibited". In our view, it is not always reasonable to assume that all actions regulated by a set of policies are prohibited, except those that are explicitly permitted. In real use cases, it may happen that certain actions are simply *possible*, i.e. they are not explicitly permitted but they are not prohibited: this because in certain situations the request of explicitly permitting certain actions may complicate the overall specification of the set of policies. This assumption requires also that permissions are stronger than prohibitions or that the prohibition to perform an action is deleted when a permission to perform such an action is added.

In March 2016 the W3C "Permissions and Obligations Expression" (POE) Working Group was created with the goal of bringing the Community Group specifications of ODRL 2.1 through the W3C Process to "Recommendation" status. In the following sections we will refer to the most recent version of the Model[3].

In our proposal, a conditional obligation is characterized by the following properties: an *assigner*, an *assignee*, a *deadline*, an *activation condition*, and a *content*. Conditional obligations could be expressed in ODRL by introducing the following syntactic extensions to the RDF Core Model.

– We introduce the class Obligation as a subclass of the ODRL class Rule and we use the property obligation: Policy → Rule for connecting a policy with an obligation.
– Like in ODRL, an obligation may have none or one assigner and/or assignee, which are specified by using the property assigner: Role → Party and the property assignee: Role → Party.
– Differently from ODRL, in our model an obligation does not need to be connected by the target property with the asset whose use is regulated by the obligation. This because the regulated asset is specified by the activation condition of the obligation. In fact, in our model actions are not simply labels like "odrl:read", but they are specified using a richer content language, actions have a class/type and a certain number of properties. In our examples

[3] A preview of the new, even if unstable, ODRL Information Model is available at https://www.w3.org/TR/odrl-model/.

we will use the RDF Schema definition of the notion of action proposed by `Schema.org` vocabulary[4].

– Moreover we introduce the following new properties:
  - hasActCond: Obligation $\rightarrow$ Action, which is used for connecting an obligation with its activation condition. The activation condition may describe a specific action or a class of actions, this last option is possible by specifying only the type of the action and some of its properties;
  - hasContent: Rule $\rightarrow$ Action, which is used for connecting an obligation to its content, that is, the description of the action that should be performed for fulfilling the obligation. This action, like for the activation condition, may be described in full details or by specifying only the type of the action and some of its properties;
  - hasDeadline: Obligation $\rightarrow$ TemporalEntity, which is used for connecting an obligation to the deadline within which the action, described in the content of the obligation, has to be performed.

Another example of conditional obligation, which is common in electronic commerce of digital assets like music or video file, is the obligation to pay to a music provider (for example Sony) a certain amount (for example 5 euros) before a given deadline (for example the end of September 2017) when an agent (for example Billy:888) plays a song recorded by the a certain artist, for example the Beatles. Obviously, this conditional obligation may be initially formalized at a high level of abstraction using roles and variables, like the role of music provider, or the role of listener or the variable used for expressing the amount to be paid. This high-level obligation may be transformed through various steps into a low-level, concrete obligation without roles and variables. The specification of a possible concrete obligation formalized in RDF using the Turtle serialization format[5] is reported below. Thanks to the choice of using a semantic web language like RDF for the specification of our example, we are able to exploit the crucial advantage of importing other ontologies in our example. In particular our RDF specification of obl:01 uses: the ODRL 2.1 RDF Ontology, the Time Ontology in OWL[6], the RDF ontology of the notion of Action defined by `schema.org`, our OWL Event Ontology[7] which defines the notion of Event and connects it with the notion of Time[5], and the OWL Normative Language Ontology that results from the previously discussed extension of ODRL[8].

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix odrl: <http://www.w3.org/ns/odrl/2/>.
@prefix schema: <http://schema.org/>.
@prefix time: <http://www.w3.org/2006/time#>.
```

[4] http://schema.org/Action.
[5] https://www.w3.org/TR/turtle/.
[6] https://www.w3.org/TR/2006/WD-owl-time-20060927/.
[7] Available at http://www.people.usi.ch/fornaran/ontology/Event.
[8] Available at http://www.people.usi.ch/fornaran/ontology/NormativeLanguage.

```
@prefix event: <http://www.people.usi.ch/fornaran/ontology/event#>.
@prefix nl: <http://www.people.usi.ch/fornaran/ontology/NormativeLanguage#>.

<http://example.com/policy:01>
    a odrl:Agreement;
    nl:obligation <http://example.com/obl:01>.
<http://example.com/obl:01>
    a nl:Obligation;
    nl:hasDeonState nl:conditional;
    odrl:assigner <http://example.com/sony:10>;
    odrl:assignee <http://example.com/billie:888>;
    nl:hasDeadline [
        time:inDateTime "2017-08-21T09:00:00"^^xsd:dateTime];
    nl:hasActCond [
        a schema:ListenAction;
        nl:hasState nl:unSatisfied;
        schema:agent <http://example.com/billie:888>;
        schema:object [
            a schema:MusicRecording;
            schema:byArtist <http://example.com/Beatles>]
    ];
    nl:hasContent [
        a schema:PayAction;
        nl:hasState nl:unSatisfied;
        schema:agent <http://example.com/billie:888>;
        schema:recipient <http://example.com/sony:10>;
        schema:price 5.00;
        schema:priceCurrency "euro"].
<http://example.com/sony:10> a odrl:Party.
<http://example.com/billie:888> a odrl:Party.
<http://example.com/Beatles> a schema:MusicGroup.
```

4 Operational Semantics of Obligations

One relevant limit of ODRL 2.1 is that it has only an informal semantics described in English in the Core Model document and it there is not a formal specification of its semantics [19]. Without a formal semantics, it is not possible to perform automatic reasoning on those policies and therefore it is almost impossible to automatically provide relevant services. One important service is *monitoring* of obligations, in order to automatically compute their activation, and fulfilment or violation on the basis of the (usually partial) knowledge of the actions of the parties involved and of the state of their interaction. Monitoring techniques together with a generator of events and/or the list of actions that one or more users would like to perform on a set of digital assets, can be used to *simulate* the evolution of the state of obligations and then perform a "*what if*" analysis on their fulfilment or violation. This is also a possible approach for evaluating norms *conflicts* or norms compatibility that is thought to work for obligations with a dynamic evolution from active to fulfilled or violated.

A concrete use case where the simulation service proposed may be used is to assist a user who wants to use different digital assets (some photos, one video, and certain data) for creating a conference presentation and then publish it on the Internet on a Social Network. The user wants to know if a given sequence of actions performed on the digital assets will bring to a violation of one of the policies connected to the digital assets used and, if the answer is in the positive, what are the reasons of the violation.

In this section, we present an operational semantics of the syntactic extension of ODRL for expressing conditional obligations, described in the previous section. In particular, the proposed semantics will be focused on the goal of performing automatic reasoning on the temporal evolution of obligations on the basis of the actions of the involved parties and of the elapsing of time. The first step, for the definition of such an operational semantics, consists in the unambiguous specification of the lifecycle of conditional obligations, which depends on the satisfaction of their condition and content. Figure 1 shows the lifecycle of conditional obligations. In this figure, boxes indicate states of the obligation, arrows indicates transitions between states, labels on arrow indicate the preconditions that must be satisfied for the transition to take place or the action that determines the transition.

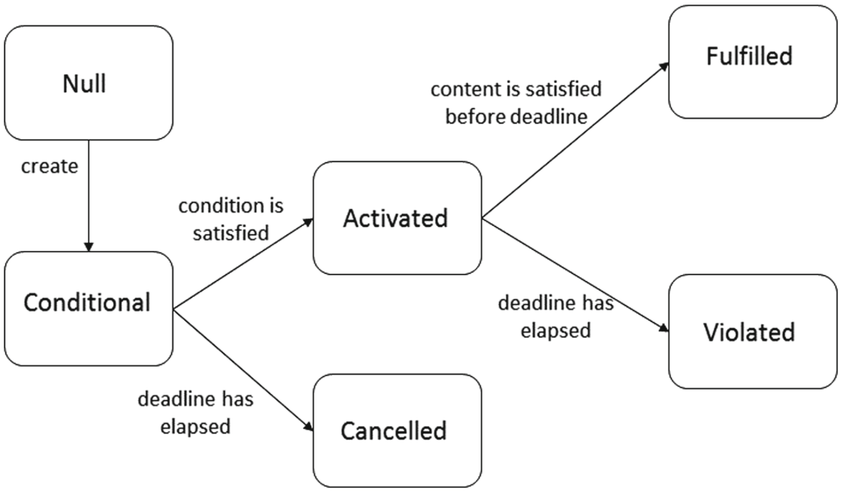

Fig. 1. Conditional obligations lifecycle

When an obligation is created, it becomes *conditional* if it has an unsatisfied condition. A *conditional* obligation becomes *activated* when its condition is satisfied (e.g., the song is played). The condition or the content of obligations becomes satisfied if, in the ontology used for representing the state of the interaction, there is an action whose attribute values match the values specified in the condition or content of the obligation. The condition or the content of an obligation is connected to the matched action or event by means of the reason property. An *active* obligation becomes *fulfilled* if its content becomes satisfied before the deadline. On the contrary, if the deadline of an active obligation has elapsed the obligation becomes *violated*.

The second step for the definition of the operational semantics of conditional obligations consists in specifying how the dynamic evolution of the various obligations can be automatically computed, in order to provide *monitoring* and *simulation* services.

Our approach in this phase is to adopt, as far as possible, W3C standards and in particular Semantic Web Technologies. This first of all because their diffusion among software practitioners and research communities is fundamental for the acceptance and adoption of the proposed formal semantics, and secondly because W3C Semantic Web Technologies are supported by tools (RDFS and OWL reasoners, editors, and API) that are suitable to implement a software framework for testing and evaluating our proposal. Moreover, in this paper we propose an extension of ODRL Version 2.1 Core Model, which is available in RDF 1.0. This choice makes it also feasible to re-use existing ontologies as discussed in the previous section, and to express policies that depend on the semantic content of the data when these are represented using Semantic Web Technologies, like for example Linked Open Data.

An open question at this point is whether it is a good solution to use OWL 2 (possibly integrated with SWRL whenever the expressivity of OWL is insufficient) to express the state of the interaction (i.e. the actions performed by the agents), the obligations and the axioms/rules for computing the state of obligations following their lifecycle. Alternatively, it may be better to express our data (state and obligations) and the obligations' lifecycle using RDFS ontologies, and exploit an RDFS reasoner, like the Apache Jena RDFS reasoner, for computing the state of the obligations. The latter option would allow us to integrate the Apache Jena RDFS reasoner with domain-specific rules, written in the rule language of the Jena general purpose rule engine[9], which can be used to represent the lifecycle of obligations.

In our previous works [5,16] we proposed to express the content and the activation condition of norms by using OWL classes defined by means of OWL axioms and, whenever the expressivity of OWL was insufficient, by SWRL rules. At runtime, the satisfaction of such activation condition and content can be evaluated using a standard OWL reasoning service or SPARQL-DL queries [7]. The main problem that we tackled with such an approach concerns performance, mainly due to the time required for reasoning on OWL ontologies when the number of assertions in the data ontology and the number of norms increase.

Therefore, in the work described in this paper we investigated the expressivity and performance of a solution where the state of the interaction and the obligations' components are formalized using RDFS and the state of obligations is computed by using the Apache Jena general-purpose rule engine. This rule engine integrates a general purpose rule-based reasoner, able to implement

[9] Documentation available at https://jena.apache.org/documentation/inference/.

RDFS and OWL reasoning, but is also available for general use[10]. In particular, we propose to exploit the Jena forward chaining RETE engine for reasoning on the state of the obligations formalized in RDFS using the RDFS Schema proposed in the previous section.

The Jena rules that can be used for computing if an obligation is cancelled, activated, fulfilled or violated are the following.

```
[cancelObligation:
(?obl   rdf:type    nl:Obligation),
(?obl   nl:hasDeonState   nl:conditional),
(?obl   nl:hasDeadline   ?deadLine),
(?deadLine time:inDateTime ?dateTime),
now(?now),
greaterThan(?now,?dateTime)      ->
(?obl   nl:hasDeonState   nl:cancelled),
drop(1)
]

[activateObligation:
(?obl   rdf:type    nl:Obligation),
(?obl   nl:hasDeonState   nl:conditional),
(?obl   nl:hasActCond ?activation),
(?activation nl:hasState nl:satisfied) ->
drop(1),
(?obl   nl:hasDeonState  nl:activated)
]

[fulfillObligation:
(?obl   rdf:type    nl:Obligation),
(?obl   nl:hasDeonState   nl:activated),
(?obl   nl:hasDeadline   ?deadLine),
(?deadLine time:inDateTime ?dateTimeDeadline),
(?obl    nl:hasContent  ?content),
(?content nl:hasState nl:satisfied),
(?content nl:reason ?instance),
(?instance event:atTime ?instant),
(?instant time:inDateTime ?dateTimeInstance),
lessThan(?dateTimeInstance,?dateTimeDeadline)  ->
(?obl   nl:hasDeonState   nl:fulfilled),
drop(1)
]
```

[10] More precisely "there are two internal rule engines one forward chaining RETE engine and one tabled datalog engine - they can be run separately or the forward engine can be used to prime the backward engine which in turn will be used to answer queries.".

```
[violateObligation:
(?obl   rdf:type    nl:Obligation),
(?obl   nl:hasDeonState   nl:activated),
(?obl   nl:hasDeadline   ?deadLine),
(?deadLine time:inDateTime ?dateTime),
now(?now),
greaterThan(?now,?dateTime)   ->
(?obl   nl:hasDeonState   nl:violated),
drop(1)
]
```

In order to compute the satisfaction of the condition and of the content of obligations we need to introduce had-hoc rules for every obligation. These rules change on the basis of the attributes specified in the description of the actions or events in the content and condition of obligations. For example the rule for computing the satisfaction of the condition of obligation `obl:01` specified in the previous section is:

```
[satisfyCondObl01:
(<http://example.com/obl:01> nl:hasDeonState   nl:conditional),
(<http://example.com/obl:01> nl:hasActCond ?activation),
(?activation rdf:type ?actClass),
(?activation nl:hasState nl:unSatisfied),
(?activation schema:agent ?agent),
(?activation schema:object ?object),
(?object rdf:type ?objectClass),
(?object schema:byArtist ?artist),
(?inst  rdf:type ?actClass),
(?inst  schema:agent ?agent),
(?inst  schema:object ?object1),
(?inst  event:atTime ?instant),
(?instant time:inDateTime ?dateTime),
now(?now),
greaterThan(?now,?dateTime),
(?object1  rdf:type ?objectClass),
(?object1 schema:byArtist ?artist)   ->
(?activation nl:hasState nl:satisfied),
(?activation nl:reason ?inst),
drop(3)
]
```

5 Evaluations and Conclusions

We have implemented a Java prototype of the proposed approach. We formalized policies containing conditional obligations using RDF 1.0, and computed their state based on a RDF 1.0 representation of the actions performed by the parties involved in the obligations. The state of conditional obligations and the satisfaction of their condition and content have been computed using forward rules formalized for the Jena general-purpose rule engine. Tests were made with a PC with an Intel Core i5-6600 CPU 3.3 GHz processor, 8 GB RAM. The computation of the satisfaction of the content and condition of one obligation and of its state requires 555 ms. The computation time becomes 557 ms with 10 policies (having all the same content and condition), 569 ms with 20 policies, and 576 ms with 30 policies. From this experiment, we can conclude that the number of policies (and in our case the number of obligations) does not significantly influence the computation time.

We intend to continue this work by proposing an extension and a syntax of the ODRL notion of permission and prohibition. We plan to study how it is possible to integrate efficiently the automatic reasoning on the state of the obligations with the reasoning on the semantics of their content and conditions. This could be done by exploiting the possibility to combine the Jena RDFS reasoner with custom rules (like the one proposed in this paper) or by using one of the available external reasoners, like for example Pellet [14]. We plan also to investigate the possibility to express the activation condition of obligations based on the semantics of the protected digital assets and to study the process for automatically transforming high-level policies expressed in terms of roles and variables into concrete policies.

References

1. Abelson, H., Creative Commons (Organization): CcREL: The Creative Commons Rights Expression Language (2008)
2. Becker, S., Hück, B., Naujokat, K., Schmeiser, A.F., Kasten, A.: ODRL 2.0 revisited. In: Horbach, M. (ed.) GI-Jahrestagung. LNI, vol. 220, pp. 3081–3095. GI (2013)
3. Bradshaw, J.M., et al.: The KAoS Policy Services Framework. In: Eighth Cyber Security and Information Intelligence Research Workshop (CSIIRW 2013), Oak Ridge, p. 2013. Oak Ridge National Labs, TN (2013)
4. da Silva Figueiredo, K., Torres da Silva, V., de Oliveira Braga, C.: Modeling norms in multi-agent systems with NormML. In: De Vos, M., Fornara, N., Pitt, J.V., Vouros, G. (eds.) COIN -2010. LNCS (LNAI), vol. 6541, pp. 39–57. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21268-0_3
5. Fornara, N.: Specifying and monitoring obligations in open multiagent systems using semantic web technology. In: Elçi, A., Koné, M.T., Orgun, M.A. (eds.) Semantic Agent Systems: Foundations and Applications. Studies in Computational Intelligence, vol. 344, pp. 25–46. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18308-9_2
6. Fornara, N., Colombetti, M.: Representation and monitoring of commitments and norms using OWL. AI Commun. **23**(4), 341–356 (2010)
7. Fornara, N., Marfia, F.: Modeling and enforcing access control obligations for SPARQL-DL queries. In: Fensel, A., Zaveri, A., Hellmann, S., Pellegrini, T. (eds.) Proceedings of the 12th International Conference on Semantic Systems, SEMANTICS 2016, Leipzig, Germany, 12–15 September 2016, pp. 145–152. ACM (2016)
8. Governatori, G., Rotolo, A.: BIO logical agents: norms, beliefs, intentions in defeasible logic. Auton. Agents Multi-Agent Syst. **17**(1), 36–69 (2008)

9. Governatori, G., Rotolo, A., Villata, S., Gandon, F.: One license to compose them all. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 151–166. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_10
10. Iannella, R., Guth, S., Paehler, D., Kasten, A.: ODRL Version 2.1 Core Model (2015). https://www.w3.org/community/odrl/model/2.1/. Accessed 15 Sept 2017
11. Jain, P., Hitzler, P., Janowicz, K., Venkatramani, C.: There's no money in linked data (2013)
12. Kagal, L., Finin, T.W., Joshi, A.: A policy language for a pervasive computing environment. In: 4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), 4–6 June 2003, Lake Como, Italy, p. 63. IEEE Computer Society (2003)
13. Kasten, A., Grimm, R.: Making the semantics of ODRL and URM explicit using web ontologies. In: Virtual Goods, pp. 77–91 (2010)
14. Meditskos, G., Bassiliades, N.: Dlejena: a practical forward-chaining OWL 2 RL reasoner combining jena and pellet. J. Web Sem. **8**(1), 89–94 (2010)
15. Moses, T.: Extensible access control markup language (xacml) version 2.0. OASIS Standard, 200502 (2005)
16. Nguyen, T.T., Fornara, N., Marfia, F.: Automatic policy enforcement on semantic social data. Multiagent Grid Syst. **11**(3), 121–146 (2015)
17. Rodríguez-Doncel, V., Villata, S., Gómez-Pérez, A.: A dataset of RDF licenses. In: Hoekstra, R. (ed.) Legal Knowledge and Information Systems - JURIX 2014: The Twenty-Seventh Annual Conference. Jagiellonian University, Krakow, Poland, 10–12 December 2014. Frontiers in Artificial Intelligence and Applications, vol. 271, pp. 187–188. IOS Press (2014)
18. Sensoy, M., Norman, T.J., Vasconcelos, W.W., Sycara, K.P.: OWL-POLAR: a framework for semantic policy representation and reasoning. J. Web Sem. **12**, 148–160 (2012)
19. Steyskal, S., Polleres, A.: Towards formal semantics for ODRL policies. In: Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., Roman, D. (eds.) RuleML 2015. LNCS, vol. 9202, pp. 360–375. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21542-6_23
20. Uszok, A., et al.: New developments in ontology-based policy management: Increasing the practicality and comprehensiveness of kaos. In: 9th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2008), 2–4 June 2008, Palisades, New York, USA, pp. 145–152. IEEE Computer Society (2008)