Towards a Formal Semantics of the Open Digital Rights Language (ODRL 2.2)

Piero Andrea Bonatti[1,†], Nicoletta Fornara[2,*,†] and Andreas Harth[3,†]

[1]University of Naples "Federico II", Napoli, Italy
[2]Università della Svizzera italiana, Lugano, Switzerland
[3]Friedrich-Alexander-Universität Erlangen-Nürnberg & Fraunhofer Institute for Integrated Circuits IIS, Nuremberg, Germany

Abstract

The Open Digital Rights Language (ODRL 2.2), being a W3C Recommendation since 2018, is a popular candidate for a policy language applied in data spaces. However, ODRL does not yet have a formal semantics. In this paper, we address this issue by making the first steps towards a formal, declarative semantics for ODRL. We start by covering a subset of the language that includes only ODRL's main features (permissions, prohibitions, obligations, duties, refinements and constraints). This exercise naturally reveals "gray areas" in ODRL's informal semantics, as well as lack of expressiveness; both will be pointed out during the formalization.

Keywords

Policy Languages, ODRL 2.2, Formal Semantics

1. Introduction

The realization of open and distributed ecosystems in which organizations, individuals, and digital companions can exchange data or resources is becoming increasingly important, as shown by the studies of various W3C Community Groups and data space initiatives supported by the European Commission[1]. Prerequisites for the realization of such systems are: (i) the sovereignty of participants over their resources, i.e., the ability to define rules for accessing and using these resources; (ii) the possibility of having interactions based on agreements and policies; (iii) the ability to verify the compliance of interactions with respect to the agreements and policies stipulated and also with respect to legislation. Central to building such systems is a language for specifying policies that can express, at least, permissions, prohibitions, and obligations for the various actions that can be performed on digital resources.

The Open Digital Rights Language (ODRL) is a policy language that can be applied to a broad variety of scenarios. Originally designed for Digital Rights Management (DRM) applications, ODRL is now being used to express manifold contracts and policies, such as data licensing [1], privacy policies [2], and access control policies [3]. ODRL is machine readable, so the various computational tasks that apply to policies (such as validation, enforcement, monitoring, compliance checking, and explanation, just to name a few) can in principle be performed by algorithms – an essential feature in scenarios that require the processing of large numbers of policies.

However, ODRL 2.2 [4, 5] is currently lacking formal semantics, which jeopardizes several of the intended applications. In general, it is essential that all the parties that have to deal with a policy understand the policy in the same way. Otherwise, their actions may result in policy violations and possibly sanctions. In particular, the party that states a policy and the party that shall comply with it need to share a common understanding of the meaning of the policy. Moreover, all the algorithms

ODRL and beyond: Practical Applications and chaLlenges for policy-base access and usage control. OPAL 2025. Co-located with the Extended Semantic Web Conference, Portorož, Slovenia · June 1, 2025.

*Corresponding author.
†These authors contributed equally.
pab@unina.it (P. A. Bonatti); nicoletta.fornara@usi.ch (N. Fornara); andreas.harth@fau.de (A. Harth)
0000-0003-1436-5660 (P. A. Bonatti); 0000-0003-1692-880X (N. Fornara); 0000-0002-0702-510X (A. Harth)

[1]https://digital-strategy.ec.europa.eu/en/policies/data-spaces

that implement the computational tasks on policies have to yield coherent results (e.g., access control shall behave like explanations, and access control decisions should not be invalidated during auditing). Unfortunately, so far, ODRL's semantics has been specified in natural language and is ambiguous in several respects, as will be pointed out later. Consequently, the above natural desiderata cannot be met.

The W3C ODRL Community Group is aware of the lack of a formal semantics for ODRL and has published a draft report on ODRL semantics [6]. Other papers in the literature have highlighted and discussed this problem. One is [7], where formal semantics of ODRL 2.1 policy expressions is proposed, with a focus on the consideration of explicit and implicit dependencies between actions. Another is [8], where an extension of the syntax of ODRL for expressing conditional obligations is proposed together with an operational semantics of such an extension based on production rules. In [9] the problem of verifying the compliance of business processes with regulatory obligations is addressed by proposing an ODRL profile and presenting a translation of ODRL policies into Answer Set Programming for compliance verification purposes.

In this paper, we propose a declarative semantics for ODRL 2.2. We start by covering a subset of the language that includes only ODRL's main features (permissions, prohibitions, obligations, duties, refinements and constraints). This exercise naturally reveals "gray areas" in ODRL's informal semantics, as well as lack of expressiveness; both will be pointed out during the formalization. We are going to formalize ODRL's semantics in a declarative, model-theoretic style, because such an approach is implementation-independent and – more importantly – *independent from the computational tasks* (that is, policy *use*), while procedural approaches are intrinsically tied to specific tasks. Declarative semantics can be used as a universal reference point to prove that all procedural solutions to any of the computational tasks conform to the same meaning of the policy. Thus, declarative semantics is the key to achieving the coherent behavior across all the different policy-related computational tasks that we are advocating. We are going to illustrate and justify our declarative semantics by means of standard examples that can be found in ODRL's specification documents.

The paper is structured as follows. Section 2 presents some examples of ODRL policies. Section 3 defines a formal declarative semantics for ODRL's main constructs, and points out several issues affecting the informal semantics. Section 4 illustrates the declarative semantics by formalizing the examples of Section 2. Section 5 illustrates how the declarative semantics provides correctness criteria for the implementation of policy-related computational tasks. Section 6 is devoted to conclusions.

2. Examples of Policies

In this section we illustrate several examples of ODRL policies. Let us start with the involved parties. Consider two people, Alice and Bob, and the organization ACME. Their URIs are `/party#Alice`, `/party#Bob`, and `/party#ACME`, respectively.

Now moving on to the digital resources, called assets in ODRL terminology.. ACME runs a web server at `http://acme.example.org/`. The web server hosts a document `/doc.html` and a music library at `/music/`. The music library contains individual songs, e.g., `/music/1999.mp3`. In addition, there is a photo album at `/photoAlbum`.

Finally, we represent the actions that parties can perform on assets. The actions we consider are listed in order of appearance and with the comments from the ODRL vocabulary[2]:

- `odrl:print`, i.e., "To create a tangible and permanent rendition of an Asset."
- `odrl:play`, i.e., "To create a sequential and transient rendition of an Asset."
- `odrl:compensate`, i.e., "To compensate by transfer of some amount of value, if defined, for using or selling the Asset."
- `odrl:archive`, i.e., "To store the Asset (in a non-transient form)."

All of these actions are special cases of `odrl:use`, i.e., "actions that involve general usage by parties". None of them is `odrl:transfer`, i.e., "actions that involve in the transfer of ownership to third parties".

[2] https://www.w3.org/ns/odrl/2/

2.1. Permissions Without Duty

Suppose that ACME allows everyone to print `/doc.html` in 300 dpi, but only before the year 2018. In ODRL terms, the policy specifies a permission rule for the class of printing actions, where the action is refined with the constraint stating that the resolution should be less than or equal to 300 dpi. The permission is further constrained, that is, the permission is only granted for actions exercised before 2018-01-01. Figure 1 shows the policy `/policy/13-14#p` (similar to Examples 13 and 14 in [4]) in a Turtle serialization.

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

</policy/13-14#p> a odrl:Policy ;
    odrl:permission [
        a odrl:Rule , odrl:Permission ;
        odrl:assigner </party#ACME> ;
        odrl:target </doc.html> ;
        odrl:action [
            a odrl:Action ;
            rdf:value odrl:print ;
            odrl:refinement [
                a odrl:Constraint ;
                odrl:leftOperand odrl:resolution ;
                odrl:operator odrl:lteq ;
                odrl:rightOperand 300 ;
                odrl:unit "dpi"
            ]
        ] ;
        odrl:constraint [
            a odrl:Constraint ;
            odrl:leftOperand odrl:dateTime ;
            odrl:operator odrl:lt ;
            odrl:rightOperand "2018-01-01"^^xsd:date
        ]
    ] .
```

Figure 1: Permission with a temporal constraint to perform a `odrl:print` action with refinement.

Clearly, the following action should be permitted by the policy:
Action 1: *Alice prints `/doc.html` with 300 dpi on 2017-12-31T14:35:27+01:00.*

2.2. Permission With Duties

Next, consider a policy in which ACME allows Bob to play the target asset `/music/1999.mp3`, but only in combination with a compensation of EUR 5.00. In ODRL terminology, the policy contains a permission rule for the play action class with a duty of the compensate action class refined by a constraint. Figure 2 shows the policy `/policy/22#p` (similar to Example 22 in [4]).

The combination of the following two actions should comply with the policy:
Action 2: *Bob compensates EUR 5.00 at datetime 2024-12-31T14:33:42+01:00.*
Action 3: *Bob plays the song `/music/1999.mp3` at datetime 2024-12-31T14:35:27+01:00.*

2.3. Prohibition

Finally consider a policy in which ACME prohibits Bob from archiving the target asset `/photoAlbum` before the year 2024. In ODRL terms, the policy consists of a prohibition rule with a constraint that the

```
@prefix odrl: <http://www.w3.org/ns/odrl/2/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

</policy/22#p> a odrl:Policy ;
    odrl:permission [
        a odrl:Rule , odrl:Permission ;
        odrl:assigner </party#ACME> ;
        odrl:assignee </party#Bob> ;
        odrl:target </music/1999.mp3> ;
        odrl:action odrl:play ;
        odrl:duty [
            a odrl:Duty ;
            odrl:action [
                a odrl:Action ;
                rdf:value odrl:compensate ;
                odrl:refinement [
                    a odrl:Constraint ;
                    odrl:leftOperand odrl:payAmount ;
                    odrl:operator odrl:eq ;
                    odrl:rightOperand 5.00 ;
                    odrl:unit "Euro"
                ]
            ]
        ]
    ] .
```

Figure 2: Permission to exercise `odrl:play` constrained by a duty to `odrl:compensate`.

datetime has to be equal to the year 2024. See Figure 3 for the policy `/policy/19#p` (similar to the example in Section 5.1 [6]).

```
@prefix odrl: <http://www.w3.org/ns/odrl/2/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

</policy/19#p> a odrl:Policy ;
    odrl:prohibition [
        a odrl:Rule , odrl:Prohibition ;
        odrl:assignee </party#Bob> ;
        odrl:assigner </party#ACME> ;
        odrl:target </photoAlbum> ;
        odrl:action odrl:archive ;
        odrl:constraint [
            a odrl:Constraint ;
            odrl:leftOperand odrl:dateTime ;
            odrl:operator odrl:lt ;
            odrl:rightOperand "2024-01-01"^^xsd:date
        ]
    ] .
```

Figure 3: Prohibition for Bob to perform `odrl:archive` on `/photoAlbum` before 2024.

Accordingly, the following action does not comply with the policy:
Action 4: *Bob archives* `/photoAlbum` *on 2023-06-18.*

3. Formalization

In order to propose the following formalization of the semantics of ODRL 2.2 we considered the specification written in Natual Language available in the ODRL Information Model 2.2 [4] and in the ODRL Vocabulary & Expression 2.2 [5].

3.1. Entities and Datatypes

We assume m *domains* $\Delta_1, \ldots, \Delta_m$, each of which contains real-world entities of a particular class or data type used in the ODRL policy (like assets, parties, numbers, dates, etc.). Note that m may be larger than the number of classes and data types mentioned in ODRL's information model and core vocabulary, because an ODRL profile may add further entity types to the core ones. In the following, the domains of assets, actions, and parties will be denoted by Δ_{as}, Δ_{ac}, and Δ_{pa}, respectively.

Note that both assets and parties may be either collections (i.e. sets) or individual objects. In order to simplify technical definitions in the rest of the paper, we introduce a function "set " that transforms all assets and parties into sets, for uniformity. Formally,

$$\mathsf{set}(x) = \begin{cases} x & \text{if } x \text{ is a set} \\ \{x\} & \text{otherwise.} \end{cases}$$

For each domain $\Delta_i \neq \Delta_{\text{ac}}$, there exists a corresponding set of *identifiers* $\mathbf{I}_i$ that represent the elements of Δ_i according to the serialization(s) prescribed by ODRL's (profile) specification. For example, if Δ_i is the set of all integers, then $\mathbf{I}_i$ is the set of expressions that represent integers, such as `{"@value": "1200", "@type": "xsd:integer"}`. Two or more expressions may denote the same element, like `"2002-05-30T09:30:10Z"` and `"2002-05-30T08:30:10-01:00"`, which represent the same date and time using different time zones. We assume that the sets $\mathbf{I}_i$ are mutually disjoint. Moreover, let

$$\mathbf{I} = \bigcup_{i=1}^{m} \mathbf{I}_i \text{ and } \Delta = \bigcup_{i=1}^{m} \Delta_i \,.$$

The meaning of identifiers is specified by a *denotation function*

$$\delta : \mathbf{I} \to \Delta \,,$$

such that for each $i \in [1, m]$, and for all $c \in \mathbf{I}_i$, $\delta(c) \in \Delta_i$. For example, if

$$c = \text{"}\texttt{http://acme.example.org/music/1999.mp3}\text{"},$$

then $\delta(c)$ denotes the actual (physical) mp3 file located at c.

The domain Δ_{ac} of action *instances* is structured in a slightly different way. The action terms ("use", "transfer", "play", etc.) denote *classes* of action instances, that is, for each action term a,

$$\delta(a) \subseteq \Delta_{\text{ac}} \,,$$

and whenever ODRL's core ontology or a profile ontology contain the expression a `odrl:includedIn` b, then $\delta(a) \subseteq \delta(b)$. For example $\delta(\text{"}\texttt{odrl:print}\text{"}) \subseteq \delta(\text{"}\texttt{odrl:use}\text{"})$.

Furthermore, two meta-functions party and obj associate each action instance $a \in \Delta_{\text{ac}}$ with the set of actors that exercise it, and the set of assets involved. Formally, for all $a \in \Delta_{\text{ac}}$,

$$\mathsf{party}(a) \subseteq \Delta_{\text{pa}} \text{ and } \mathsf{obj}(a) \subseteq \Delta_{\text{as}} \,.$$

op	$\delta(\text{op})$	op	$\delta(\text{op})$
eq	$=$	isA	$\in$
gt	$\bigcup_{i=1}^{m} (>_i)$	hasPart	$\ni$
gteq	$\bigcup_{i=1}^{m} (\geq_i)$	isPartOf	$\in$
lt	$\bigcup_{i=1}^{m} (<_i)$	isAllOf	$=$
lteq	$\bigcup_{i=1}^{m} (\leq_i)$	isAnyOf	$\in$
neq	$\neq$	isNoneOf	$\notin$

Table 1
Semantics of the core operators of ODRL

deliveryChannel	$\Delta_{\text{ac}} \to \{\text{mobileNetwork}, \ldots\}$
industryContext	$\Delta_{\text{ac}} \to \{\text{publishing}, \text{financialIndustry}, \ldots\}$
paymentAmount	$\Delta_{\text{ac}} \to \mathbb{Q}$ (rational numbers)
assetPercentage	$\Delta_{\text{ac}} \to [0, 1]$
purpose	$\Delta_{\text{ac}} \to \{\text{education}, \text{nonCommercial}, \text{Marketing}, \ldots\}$
recipient	$\Delta_{\text{ac}} \to \Delta_{\text{pa}}$
resolution	$\Delta_{\text{ac}} \to \mathbb{N}$ (natural numbers)
size	$\Delta_{\text{as}} \to \mathbb{N}$
version	$\Delta_{\text{as}} \to \text{String}$

Table 2
Some nontemporal left operand types.

3.2. Constraint Operators

ODRL's core operators (used in constraints and refinements) refer to totally ordered domains (such as numbers, strings, dates) and collections, because parties and assets may be either individual entities or sets thereof. Let us denote with $\leq_i$ the ordering associated with Δ_i (if Δ_i is not ordered, then $\leq_i$ is the empty relation); then the semantics of ODRL's operators is specified in Table 1, where $>_i$, $\geq_i$, and $<_i$ are derived from $\leq_i$ in the usual way. By analogy with identifiers, we denote the meaning of an operator o with $\delta(o)$.

For example, $\delta(\text{eq})$ is the set of all pairs $(x, y) \in \Delta \times \Delta$ such that $x = y$; $\delta(\text{gt})$ is the set of all pairs $(x, y) \in \Delta \times \Delta$ such that, for some $i \in [1, m]$, $x >_i y$; $\delta(\text{isA})$ is the set of all pairs $(x, y) \in \Delta \times \Delta$ such that $x \in y$.

The meaning of the logical operators `or`, `xone`, `and` is defined similarly, using the standard truth tables of boolean connectives. For example, $\delta(\text{or})$ is the set of pairs $\{(true, true), (true, false), (false, true)\}$.

Remark 1. ODRL supports also a logical operator `andSequence`, which prescribes the evaluation of its arguments from left to right. Given that logical `and` is symmetric and that ODRL's core constraints have no side effects, it is hard to find any differences between `and` and `andSequence`. Currently there are no examples on the intended use of the latter, so its formalization is deferred until a clearer specification of its meaning will be available. □

3.3. Left Operands and World States

ODRL's constraints and refinements contain a left operand that expresses the properties of assets, parties, actions, and other ODRL entities, and possibly the properties of the environment. Table 2 lists some of the core left operands and their type; most of them specify properties of actions and assets.

Property values may change. A *state* associates properties with their values at a particular point in time and lists the events that occur at that time. Thus, states can be thought of as snapshots of the world. ODRL also supports left operands that depend on a history of states, such as "count"; these operands, which we call *temporal left operands*, will be discussed in a future paper.

Formally, let $\mathbf{LO_n} = \{\ell_1, \ldots, \ell_l\}$ be the set of nontemporal left operands, let Δ_{ev} be the domain of events (we assume that actions are events, that is, $\Delta_{\text{ac}} \subseteq \Delta_{\text{ev}}$), and let Δ_{dt} be the domain of

date-and-time points. A *state* is a triple

$$S = \langle t^S, E^S, (\cdot)^S \rangle$$

where $t^S \in \Delta_{\text{dt}}$ represents the current time, $E^S \subseteq \Delta_{\text{ev}}$ is the set of events that occur at that time, and $(\cdot)^S$ is an *interpretation function* that maps each nontemporal left operand $\ell_i \in \mathbf{LO_n}$ on a function ℓ_i^S of the appropriate type (cf. Table 2). Thus, the function $(\cdot)^S$ specifies the values of all nontemporal properties at time t^S.

3.4. Constraints (Nontemporal)

Constraints (and refinements) are expressions like "`leftOperand`:ℓ_i `operator`:op `rightOperand`:c", that we will abbreviate to $\langle \ell_i, op, c \rangle$. Here we focus on nontemporal constraints, that is, $\ell_i \in \mathbf{LO_n}$.

We say that a domain element $x \in \Delta$ *satisfies* a constraint $\langle \ell_i, op, c \rangle$ in a state S – in symbols, $S, x \models \langle \ell_i, op, c \rangle$ – iff

$$(\ell_i^S(x), \delta(c)) \in \delta(op)\,.$$

Here $\ell_i^S(x)$ is the value of x's property ℓ_i in S, $\delta(c)$ is the value denoted by the right operand, and $\delta(op)$ is the meaning of op.

Based on the above definition, the meaning of the logical constraints based on operators `or`, `xone`, `and` is defined in the obvious way.

3.5. Compliance with Permission and Prohibition Rules Without Duties and Remedies

A rule is characterized by:

- a type θ (such as "permission", "prohibition", "duty");
- an action class α (such as "use", "transfer", "play", ...);
- an optional "target" asset ob (for "object"), which is mandatory for permissions and prohibitions;
- an optional "assignee" party p (for "party");
- other optional properties such as "assigner", "duty", "remedy", and "consequence".

Before delving into the definition of compliance, we need to introduce the notion of rule activation:

Definition 1. A rule ρ is *active on* some action $a \in E^S$ in some state S iff one of the following conditions holds:

- ρ has no constraints (then ρ is trivially active);
- if ρ has a set of constraints $\langle \ell_{i,j}, op_j, c_j \rangle$ ($j = 1, \ldots, n$), then ρ is active on a iff for all $j = 1, \ldots, n$, $S, x_j \models \langle \ell_{i,j}, op_j, c_j \rangle$, where $x_j = a$ if $\ell_{i,j}$ is an action property, and $x_j = obj(a)$ if $\ell_{i,j}$ is an asset property.

In the rest of this subsection, we assume that the rule has no duties nor remedies. Duties and remedies will be dealt with in the following subsections.

Definition 2. (Compliance with permissions) We say that a *complies with* a permission π (in a state S) iff all the following conditions hold:

1. π is active on a in S;
2. $a \in \delta(\alpha)$ (the action belongs to the permitted class);
3. if π's action specification contains n refinements $\langle \ell_{i,j}, op_j, c_j \rangle$ ($j \in [1, n]$), then for all such j, $S, a \models \langle \ell_{i,j}, op_j, c_j \rangle$ (the action satisfies the refinements);
4. if the target ob is a collection, then $\mathsf{obj}(a) \subseteq \delta(ob)$; otherwise $\mathsf{obj}(a) = \{\delta(ob)\}$ (all the assets involved in a are covered by the target of π);

5. if the target has refinements $\langle \ell_{i,j}, op_j, c_j \rangle$ ($j \in [1, n]$), then for all such j, and for all $x \in \mathsf{obj}(a)$, $S, x \models \langle \ell_{i,j}, op_j, c_j \rangle$ (all the assets involved in a satisfy the refinements);
6. if an assignee p (for "peer") is specified in the permission, then $\mathsf{party}(a) \subseteq \mathsf{set}(\delta(p))$ (the actors who exercise the action are among the authorized assignees);
7. if the assignee has refinements $\langle \ell_{i,j}, op_j, c_j \rangle$ ($j \in [1, n]$), then for all such j and for all $x \in \mathsf{party}(a)$, $S, x \models \langle \ell_{i,j}, op_j, c_j \rangle$ (all the actors that exercise a satisfy the refinements).

Definition 3. (Compliance with prohibitions) If π is a prohibition, we say that a *complies with* π (in S) iff at least one of the following conditions holds:

1. π is not active on a in S;
2. $a \notin \delta(\alpha)$ (the action does not belong to the specified class);
3. π's action specification contains a refinement $\langle \ell_i, op, c \rangle$ such that $S, a \not\models \langle \ell_{i,j}, op_j, c_j \rangle$ (the action does not satisfy some refinement);
4. for all $x \in \mathsf{obj}(a)$, either $x \notin \mathsf{set}(\delta(ob))$, or there exists a target refinement $\langle \ell_i, op, c \rangle$ such that $S, x \not\models \langle \ell_{i,j}, op_j, c_j \rangle$ (none of the assets involved in the action is covered by π's target specification);
5. π specifies an assignee p, and for all $x \in \mathsf{party}(a)$, either $x \notin \mathsf{set}(\delta(p))$, or there exists a target refinement $\langle \ell_i, op, c \rangle$ such that $S, x \not\models \langle \ell_{i,j}, op_j, c_j \rangle$ (none of the parties that exercise the action satisfy π's assignee specification).

In other words, an action a is prohibited by π if a matches π's action specification, some of the assets involved in the action matches π's asset specification, and some of the parties that exercise the action match π's assignee specification.

3.6. Adding Time and Duties/Obligations

A duty is a rule with an action class α, an optional asset ob, an optional assignee p, and an optional list of constraints $\chi_1, \ldots, \chi_k$.

Definition 4. (Compliance with a duty) A state S *complies with* (or, equivalently, *fulfills*) a duty ω with the above structure iff there exists an action $a \in E^S$, called *fulfilling action for* ω, such that all of the following conditions hold:

1. ω is active on a in S (the constraints of ω are satisfied);
2. $a \in \delta(\alpha)$ (the action has the required type);
3. if ω's action specification contains n refinements $\langle \ell_{i,j}, op_j, c_j \rangle$ ($j \in [1, n]$), then for all such j, $S, a \models \langle \ell_{i,j}, op_j, c_j \rangle$ (the action satisfies the refinements);
4. if a target ob is specified in ω, then $\mathsf{obj}(a)$ equals the set of all assets $x \in \Delta_{\text{as}}$ such that (i) $x \in \mathsf{set}(\delta(ob))$ and (ii) if the target has n refinements $\langle \ell_{i,j}, op_j, c_j \rangle$ ($j \in [1, n]$), then for all such j, $S, x \models \langle \ell_{i,j}, op_j, c_j \rangle$ (the assets involved in a are exactly those specified in ω with "target" and its refinements);
5. if an assignee p is specified in ω, then $\mathsf{party}(a)$ equals the set of all parties $x \in \Delta_{\text{pa}}$ such that (i) $x \in \mathsf{set}(\delta(p))$ and (ii) if the target has n refinements $\langle \ell_{i,j}, op_j, c_j \rangle$ ($j \in [1, n]$), then for all such j, $S, x \models \langle \ell_{i,j}, op_j, c_j \rangle$ (the parties that exercise a are exactly those specified in ω with "assignee" and its refinements).

Remark 2. Different semantics for fulfillment are possible. In the above version, the action must be exercised jointly by all the specified assignees (as in: *the contract shall be approved by all assignees*). Alternatively, one might require that the action be exercised by *any* of the specified assignees (as in: *the payment shall be made by any of the assignees*), or that each assignee should independently fulfil the duty (e.g. *all assignees shall register*). Each of these three semantics is useful in different use cases. Unfortunately, ODRL does not specify which of the above semantics is the intended one, and it is not expressive enough to select the desired semantics, based on the use case. □

The duties of permission rules are obligations that must be fulfilled as a prerequisite *before* the permission is exercised. Thus, some notion of time shall be introduced in the semantics.

Definition 5. A *trace* is a sequence of states $\tau = \langle S_1, S_2, \ldots, S_z \rangle$ such that $t^{S_i} < t^{S_{i+1}}$ $(i = 1, \ldots, z)$, that is, timestamps are increasing.

Definition 6. (Compliance with a permission with duties) Given a trace $\tau = \langle S_1, S_2, \ldots, S_z \rangle$, a state S_i ($i \in [1, z]$) and and action $a \in E^{S_i}$, we say that a *complies* (in τ and S_i) with a permission π with duties $\omega_1, \ldots, \omega_d$ iff the following two conditions hold:

1. a complies with π in S_i as specified in the previous section for duty-less permissions;
2. for each duty ω_i ($i \in [1, d]$) there exists $j \in [1, i]$ such that S_j fulfills ω_i.

Remark 3. Different notions of compliance with permissions with duties are possible. In the above version, a single duty fulfillment executed in one state S_j suffices to permit any number of actions $a_1, \ldots, a_k$ exercised in states $S_{j+1}, S_{j+2}, \ldots, S_z$. This can be useful to say, for example, that *after registering to a web site (once and for all), a user may access all the contents of the web site.* In other use cases, one may need to say that *each* action execution requires a different fulfillment of the duty; this would be useful to specify *pay-per-view* policies. ODRL is not expressive enough to distinguish these two meanings – nor does it specify which of the two semantics should be adopted. □

Remark 4. The ODRL Information Model specification states that a duty is "an agreed Action that MUST be exercised (as a pre-condition to be granted the Permission)" and that "the duty property asserts a pre-condition between the Permission and the Duty". Accordingly, fulfillment has been formalized in Def. 6 by requiring that the duty is fulfilled before exercising the permitted action (cf. the condition $j \in [1, i]$). However, this notion of fulfillment cannot model duties that occur in the future, as in the norm: "*personal data can be collected but it must be deleted within one month*". A possible alternative interpretation of the specification is that the assignee(s) must *agree* to fulfill the duty before exercising the permission (the duty may be fulfilled later); however, such agreement is not formalized in the specification, therefore also this alternative interpretation needs clarifications. □

Remark 5. Sections 2.6.3 and 2.6.4 of ODRL's Information Model state that "A duty [an obligation, respectively] is fulfilled *if all constraints are satisfied* and if its action, with all refinements satisfied, has been exercised". The above definitions formalize this statement. There exists, however, another way of intending the role of constraints in duties; it could be stipulated that the duty is active – and its action shall be exercised – *only if the constraints are satisfied* (otherwise, the duty can essentially be ignored). Some of the examples in the ODRL Information Model are apparently using the latter meaning rather than the specified semantics (cf. Example 22). □

3.7. Adding Remedies to Prohibitions

Prohibitions may have "remedies", which are duties that – if fulfilled after the prohibition has been infringed – restore the prohibition state to "not infringed". In formal terms:

Definition 7. Given a trace $\tau = \langle S_1, S_2, \ldots, S_z \rangle$, a state S_i ($i \in [1, z]$) and and action $a \in E^{S_i}$, we say that a *complies* (in τ and S_i) with a prohibition π with remedies $\omega_1, \ldots, \omega_d$ iff at least one of the following conditions hold:

1. a complies with π in S_i as specified in the previous sections for remedy-less prohibitions;
2. for each remedy ω_i ($i \in [1, d]$) there exists $j \in [i, z]$ such that S_j fulfills ω_i.

Note that if the prohibition is infringed at S_i and the remedies occurs later at some S_j with $j \in [i+1, z]$, then a does *not* comply with π in all the traces $\langle S_1, \ldots, S_k \rangle$ with $k \in [i, j-1]$, while a *does* comply with π in all the traces $\langle S_1, \ldots, S_k \rangle$ such that $k \in [j, z]$. In this way the formal semantics models the scenarios where the prohibition is infringed for some time, until the remedy is fulfilled.

Remark 6. In the full specification, a duty ω that occurs in an obligation or permission rule may have "consequences", that is, further duties that restore compliance if ω has not been fulfilled. The semantics of consequences can be modelled by analogy with the semantics of remedies. The definition is easy and left to the reader. □

3.8. Complying With a Policy

An ODRL policy Π may contain one or more rules (permissions, prohibitions, and duties, possibly of different types) plus several properties for specifying namespaces and profiles, and a property "conflict" with possible values "perm" (permissions have precedence), "prohibit" (prohibitions have precedence), "invalid" (conflicts generate an error); the latter is the default value.[3]

Remark 7. ODRL cannot express gap resolution strategies, where "gap" means that an action is neither permitted nor prohibited. There are at least two standard approaches to resolving policy gaps in data security: the *open* and the *closed* default policies, that permit and deny the action, respectively. In the absence of references to gap resolution in ODRL's specification, here we will (temporarily) adopt the most conservative approach, namely, the closed default policy. The reader may easily verify that prohibitions still make sense when they partially overlap permissions, unless the conflict resolution strategy is "perm"; in that case, prohibitions are irrelevant and useless. We recommend to extend ODRL with a policy property that specifies the gap resolution strategy; it might be called "default", with values "permit", "prohibit" and "invalid". □

Definition 8. Consider a policy Π with permissions $\pi_1, \ldots, \pi_p$, prohibitions $\rho_1, \ldots, \rho_d$, obligations $\omega_1, \ldots, \omega_o$, and conflict value s (for "strategy"). For all traces $\tau = \langle S_1, S_2, \ldots, S_z \rangle$, we say that: τ *complies* with Π iff all of the following conditions hold:

1. for all obligations ω_i ($i \in [1, o]$) there exists a state S_j in τ ($j \in [1, z]$) such that S_j fulfills ω_i (i.e. all obligations are fulfilled); in the following conditions, let a_i denote a fulfilling action for ω_i;
2. for all states S_j in τ ($j \in [1, z]$) and all actions $a \in E^{S_j}$, at least one of the following two conditions holds:
   a) a is either a fulfilling action a_i for an obligation ω_i of Π, or a fulfilling action for the duty of some permission π_i that is active on some action $a' \in E^{S_k}$ ($j \leq k \leq z$);
   b) there exists a permission π_i such that a complies (in S_j and τ) with π_i, and either $s =$"perm" or a complies (in S_j and τ) with all the prohibitions $\rho_1, \ldots, \rho_d$.

In other words, a trace complies with Π if all the obligations are fulfilled, and all actions are either required or permitted.[4] Point 2b deals with the latter; the existence of an applicable permission is enough if $s =$"perm" (because in this case the permission overrides any infringed prohibition), while for the other strategies s, a shall comply with all prohibitions. To understand point 2a, first note a crucial point:

Remark 8. The fulfilling actions of obligations and duties may have to be authorized, too, but they may fall outside the scope of the policy Π, in general. For example, a content provider may state in Π that an mp3 can be played only after fulfilling the duty of paying 2 euros online, and this action is either permitted or denied by the policy Π' of another company (the bank). This observation reveals a set of open issues: ODRL provides no means to define the scope of a policy (i.e. which actions must comply with which policy), nor whether there are actions that are not subject to any policy and may be freely exercised. Moreover there is no guideline about the interplay of different policies, of which the provider + bank example is just one particular instance; in more complex examples, Π and Π' may "overlap" and conflicts may have to be solved. □

[3] A policy may also contain properties that factorize the common features of its rules; however this is just an abbreviation and does not require additional semantic definitions.

[4] According to the closed policy, an action is permitted only if there is an explicit permission rule, cf. Remark 7.

In order to deal with the above lack of specifications, in this first proposal we provide a temporary solution by stipulating (through point 2a) that fulfilling actions are implicitly compliant *with the policy that requests those actions*. Thus, in the above example, the 2 euros payment complies with the policy Π of the content provider (which "requests" that action as a duty when the file is played), although it may infringe the policy Π' of the bank. A cleaner solution needs a notion of policy scope.

4. Explanation of Formalization Through Specific Policy Examples

In this section, the formalization presented in Section 3 is explained by using different examples of **policy types** and **action instances** presented in Section 2.

4.1. Evaluation of Permissions Without Duties

We start by explaining the evaluation of a **first type** of policy, that is, a policy containing a permission constrained by a constraint in which the class of actions governed is refined by another constraint, for example, the policy `/policy/13-14#p` in Figure 1. The rule π_1 in the `/policy/13-14#p` is characterized by the following properties:

- the type of the rule, θ = `odrl:Permission`;
- the action class regulated by the rule, α = `odrl:print`;
- the target asset of the rule, ob = `/doc.html`.

Moreover, the rule π_1 is restricted by a constraint $\langle \ell_i, op, c \rangle$ where ℓ_i=`odrl:dateTime`, op=`odrl:lt`, and c=`"2018-01-01"`.

In order to evaluate the permission π_1, we have to consider an action and a state. Following the formalization presented in Section 3.5, we start by evaluating the rule π_1 on the action a_1 (Action 1 in Section 2.1) and the state $S = \langle t^S, \{a_1\}, \cdot^S \rangle$ whose relevant properties are:

- type of a_1 = `odrl:print`, that is, a_1 belongs to $\delta($`"odrl:print"`$)$ (which is the set of instances of `odrl:print`);
- actors = $\mathsf{party}(a_1)$ = {`/party#Alice`};
- assets = $\mathsf{obj}(a_1)$ = {`/doc.html`};
- `odrl:dateTime`$^S(a_1)$ = t^S = `2017-12-31T14:35:27+01:00`;
- `odrl:resolution`$^S(a_1)$ = `300`.

First, it is necessary to check whether rule π_1 is $active$ on action a_1. This is done by evaluating whether the action a_1 *satisfies* the constraint $\langle \ell_i, op, c \rangle$ of π_1. Indeed, it holds that

$$S, a_1 \models \langle \texttt{odrl:dateTime}, \texttt{odrl:lt}, \texttt{"2018-01-01"} \rangle$$

because the pair (`odrl:dateTime`$^S(a_1)$), $\delta($`"2018-01-01"`$)$) evaluates to

$$(\texttt{2017-12-31T14:35:27+01:00}, \texttt{2018-01-01}) \in \delta(\texttt{odrl:lt})\,.$$

Therefore, we can conclude that π_1 is $active$ on action a_1.

Subsequently, it is necessary to evaluate whether the action a_1 *complies* with the permission π_1 by checking all the following conditions:

1. π_1 is *active* on $a_1 \in E^S$, as we have just shown;
2. $a_1 \in \delta(\alpha)$, holds, because the type of a_1 equals the permitted type α (equivalently, a_1 is an instance of `odrl:print`);
3. a_1 satisfies the refinement: $S, a_1 \models \langle \texttt{odrl:resolution}, \texttt{odrl:lteq}, 300 \rangle$, i.e. it holds that (`odrl:resolution`$^S(a_1)$, `300`) = (`300`, `300`) $\in \delta($`odrl:lteq`$)$;
4. $\mathsf{obj}(a_1) = \mathsf{set}(\delta(ob)) = \{\delta(ob)\}$, i.e. {`/doc.html`} = {`/doc.html`};

5. the other conditions need not be checked because the target has no refinements and the assignee is not specified.

Given that all these conditions are satisfied, we can conclude that a_1 *complies* with the permission π_1 in state S (that is, a_1 is permitted by π_1). Since a_1 is the unique action in S and complies with the unique rule of policy `/policy/13-14#p`, which is a permission, the trace $\langle S \rangle$ complies with `/policy/13-14#p`, according to Def. 8.

4.2. Evaluation of Permissions With Duties

We continue by explaining the evaluation of a **second type** of policy, i.e. a policy containing a permission that is constrained by a duty, like `/policy/22#p` in Figure 2. The permission rule ρ_1 in `/policy/22#p` is characterized by:

- the type of the rule, θ_1 = `odrl:Permission`;
- the action class regulated by the rule, α_1 = `odrl:play`;
- the target asset of the rule, ob_1 = `/music/1999.mp3`;
- the assignee of the rule, p_1 = `/party#Bob`.

The duty rule ω_1 in `/policy/22#p` is characterized by:

- the type of the rule, θ_2 = `odrl:Duty`;
- the action class regulated by the rule, α_2 = `odrl:compensate`;

Moreover, the action of ω_1 is subject to the refinement $\langle$`odrl:payAmount`, `odrl:eq`, `5.00`$\rangle$.

Consider Action 2 (formalized by an instance a_2) in which Bob compensates 5.00 EUR and Action 3 (formalized by a_3) in which Bob plays the mp3 (in Section 2.2). Following the formalization presented in Section 3.6, we shall evaluate the policy on a sequence of states $\langle S_1, S_2 \rangle$ where S_1 complies with (i.e. fulfills) the duty ω_1, while in S_2 the permission ρ_1 to play the mp3 file is exercised. Accordingly, let us assume that in state S_1 there is the action $a_2 \in E^{S_1}$ and t^{S_1} equals the time when a_2 happens, i.e. `2017-12-31T14:35:27+01`. The relevant properties of a_2 are:

- type = `odrl:compensate`;
- actors = $\mathsf{party}(a_2)$ = {`/party#Bob`};
- $\texttt{odrl:dateTime}^{S_1}(a_2) = t^{S_1}$ = `2024-12-31T14:33:42+01:00`;
- $\texttt{odrl:payAmount}^{S_1}(a_2)$ = 5.00.

S_1 complies with (fulfills) the duty ω_1 because all the following conditions hold:

1. ω_1 is active because it has no constraints;
2. $a_2 \in \delta(\alpha_2)$, because the type of a_2 equals the required type α_2 = `odrl:compensate`;
3. ω_1's action specification contains a refinement, and a_2 satisfies the refinement because $S, a_2 \models \langle$`odrl:payAmount`, `odrl:eq`, `5.00`$\rangle$;
4. the target and the assignee of ω_1 are not specified.

Let us now check whether the permission ρ_1 applies to an action $a_3 \in E^{S_2}$, whose relevant properties are:

- type = `odrl:play`;
- actors = $\mathsf{party}(a_3)$ = {`/party#Bob`};
- assets = $\mathsf{obj}(a_3)$ = {`/music/1999.mp3`};
- $\texttt{odrl:dateTime}^{S_2}(a_3) = t^{S_2}$ = `2024-12-31T14:35:27+01:00`.

It is straightforward to demonstrate that a_3 complies with ρ_1 in S_2. This can be done by checking all the conditions for a duty-less permission as shown in the previous section. The second condition that must be checked for permissions with duties, is whether S_2 is preceded by a state S_j that fulfills the duty ω_1. As we have already pointed out, this second condition is satisfied with $j = 1$. Now, if a_2 and a_3 are the only actions in $\langle S_1, S_2 \rangle$, then this trace complies with `/policy/22#p` because a_2 is permitted by clause 2a in Def. 8 while a_3 is permitted by clause 2b.

4.3. Evaluation of Prohibitions

In this section we explain the evaluation of a **third type** of policy, i.e. a policy containing a prohibition constrained by a constraint, e.g. the policy `/policy/19#p` in Figure 3. The rule π_1 in `/policy/19#p` is characterized by:

- the type of the rule, θ = `odrl:Prohibition`;
- the action class regulated by the rule, α = `odrl:archive`;
- the target asset of the rule, ob = `/photoAlbum`;
- the assignee party of the rule, p = `/party#Bob`.

Now we check whether the prohibition π_1 forbids, in a state S, a given action $a_4 \in E^S$, which corresponds to Action 4 (formalized by an instance a_4) in Section 2.3. Following the formalization presented in Section 3.5, the relevant properties of a_4 are:

- type = `odrl:archive`;
- actors = $\mathsf{party}(a_4)$ = {`/party#Bob`};
- assets = $\mathsf{obj}(a_4)$ = {`/photoAlbum`};
- `odrl:dateTime`$^S(a_4)$ = `2023-06-18`.

Moreover, the rule π_1 is restricted by a constraint $\langle \ell_i, op, c \rangle$ where ℓ_i = `odrl:dateTime`, op = `odrl:lt`, and c = `"2024-01-01"`.

First, it is necessary to check whether π_1 is $active$ on a_4 in S. This amounts to checking whether a_4 *satisfies* the constraint of π_1. This evaluation is similar to the evaluation in Section 4.1 and we can conclude that π_1 is $active$ on action a_4.

Subsequently, it is necessary to evaluate if the action a_4 *complies* with the prohibition π_1 by checking whether at least one of the following conditions holds:

1. π_1 is *not active* on a_4 in S; as we pointed out, this condition does not hold;
2. $a_4 \notin \delta(\alpha)$; this does not hold either, because the type of a_4 equals the prohibited type α;
3. some action refinement is violated; since there are no refinements, this condition does not hold;
4. it should be $\mathsf{obj}(a_4) \cap \mathsf{set}(\delta(ob)) = \emptyset$, since there are no target refinements; however the intersection equals {`/photoAlbum`}, therefore, this condition does not hold;
5. similarly, none of the assignees should match the assignee specification of π_1; since $\mathsf{party}(a_4)$ = $\mathsf{set}(\delta(p))$ = {`/party#Bob`}, this condition is not satisfied, either.

As none of the above conditions hold, we can say that a_4 *does not comply* with the prohibition π_1, that is, Bob's action infringes the prohibition. Accordingly, the trace $\langle S \rangle$ does not comply with `/policy/19#p` (cf. Def. 8).

5. Declarative Semantics as an Algorithm Correctness Criterion

Here we illustrate how declarative semantics can be used to assess the correctness of the algorithms designed for specific computational tasks, and guarantee their mutual interoperability. First we provide an abstract definition of the main classes of such algorithms. We define also what it means for these algorithms to be correct, based on the declarative semantics (i.e., the notion of compliance defined in the previous section).

Definition 9.

1. A *monitoring mechanism* MM is an algorithm that takes as input a trace τ (which models a system log) and a policy Π and returns a boolean value (compliant/non compliant); MM is *correct* if it returns *true* if and only if τ complies with Π;

2. an *access control mechanism* AC is an algorithm that takes a trace τ (which models the current situation), a policy Π, and a request of an action a, represented as a state $S = \langle t^S, \{a\}, (\cdot)^S \rangle$,[5] and returns a boolean value; AC is *correct* if it returns *true* if and only if the trace $\tau \cdot S$ (the concatenation of τ and S) complies with Π (that is, the execution of the action does not lead to a violation of Π);
3. a *policy comparison mechanism* PC is an algorithm that takes two policies Π_1 and Π_2 and returns a boolean value; PC is correct if it returns *true* if and only if all the traces τ that comply with Π_1 comply also with Π_2.[6]

As we claimed, it is now possible to guarantee that different algorithms behave in a coherent way, based on a single, unambiguous meaning of the policies involved. For example, it follows immediately from the above definitions that correct monitoring, access control and comparison mechanisms always return mutually compatible results:

Proposition 1.

1. *If a monitoring mechanism MM and an access control mechanism AC are correct, then for all traces τ and for all action requests S,*

$$AC(\tau, \Pi, S) = MM(\tau \cdot S, \Pi)$$

   (i.e. access control decisions are never invalidated during auditing);
2. *if a monitoring mechanism MM and a policy comparison mechanism PC are correct, then for all traces τ and all policies Π_1 and Π_2, the following facts hold:*
   a) *for all traces τ, if $PC(\Pi_1, \Pi_2) = true$ and $MM(\tau, \Pi_1) = true$, then $MM(\tau, \Pi_2) = true$ (when PC says that Π_1 complies with Π_2, if auditing is successful w.r.t. Π_1, then it is successful also w.r.t. Π_2);*
   b) *if $PC(\Pi_1, \Pi_2) \neq true$ then there exists a trace τ such that $MM(\tau, \Pi_1) = true$, and $MM(\tau, \Pi_2) = false$ (when PC says that Π_1 does not comply with Π_2, there exists indeed a trace τ such that auditing is successful only for Π_1);*
3. *if an access control mechanism AC and a policy comparison mechanism PC are correct, then the following fact holds: for all traces τ and all action requests S, if $PC(\Pi_1, \Pi_2) = true$ and $AC(\tau, \Pi_1, S) = true$, then $AC(\tau, \Pi_2, S) = true$ (when PC says that Π_1 complies with Π_2, then access is granted by Π_1 only if it is granted also by Π_2).*

Note that *one* correctness proof for each algorithm suffices to guarantee that its behavior is mutually compatible with that of *all* correct algorithms. Without a unique reference semantics, compatibility could only be guaranteed by proving a result similar to one of the above three propositions *for each pair of algorithms.*

6. Conclusion

We have provided a formalization of part of the ODRL 2.2 specification as provided by the ODRL Information Model 2.2 [4] and in the ODRL Vocabulary & Expression 2.2 [5]. The formalization currently covers permissions with duties, prohibitions with remedies, and obligations, including refinements and constraints; consequences have been sketched. The formalization is currently not covering left operands such as "count", that depend on traces; these constructs will be covered in an extended version of this paper. Further, we have illustrated the behavior of the formalization through several examples

[5] Here t^S represents the time of the request, while $(\cdot)^S$ specifies the properties of a, of the involved assets and peers, and of the environment.

[6] Thus, in informal terms, a correct PC returns *true* iff Π_1 complies with Π_2, or – equivalently – Π_1 is stronger than Π_2; a third (equivalent) formulation is: Π_1 is contained in Π_2.

and pointed out several aspects that could benefit from language extensions, and from clarifications on the official ODRL specification.

Besides extending the coverage of the formalization, future work could: propose improvements of ODRL to overcome its current limitations; clarify the relationship between ODRL and existing W3C recommendations (such as XML Schema Part 2: Datatypes Second Edition [10] and RDF 1.1 Semantics [11]); further investigate policy scope, and how permissions and prohibitions are combined into policies; and extend the expressiveness of ODRL's temporal constraints.

Acknowledgments

P. A. Bonatti acknowledges financial support from the PRIN 2022 project 2022LA8XBH-POLAR, and project SERICS (PE00000014) under the NRRP MUR program funded by the EU-NGEU. A. Harth acknowledges support from IGF project GRANERGIZE (FKZ 01IF23286N).

This work was conceived during the Dagstuhl Seminar 25051 *"Trust and Accountability in Knowledge Graph-Based AI for Self Determination"*, which took place in January 2025.

Declaration on Generative AI

During the preparation of the paper, the authors used DeepL for grammar and spelling check, paraphrase, and reword. The authors reviewed and edited the text and take full responsibility for the papers's content.

References

[1] V. Rodríguez-Doncel, S. Villata, A. Gómez-Pérez, A dataset of RDF licenses, in: R. Hoekstra (Ed.), Legal Knowledge and Information Systems - JURIX 2014, Krakow, Poland, 10-12 December 2014, volume 271 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2014, pp. 187–188. doi:10.3233/978-1-61499-468-8-187.

[2] H. J. Pandit, B. Esteves, G. P. Krog, P. Ryan, D. Golpayegani, J. Flake, Data privacy vocabulary (DPV) - version 2.0, in: G. D. et al. (Ed.), The Semantic Web - ISWC 2024 - 23rd International Semantic Web Conference, Baltimore, MD, USA, November 11-15, 2024, Proceedings, Part III, volume 15233 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 171–193. doi:10.1007/978-3-031-77847-6_10.

[3] B. Esteves, V. Rodríguez-Doncel, H. J. Pandit, N. Mondada, P. McBennett, Using the ODRL Profile for Access Control for Solid Pod Resource Governance, in: P. G. et al. (Ed.), The Semantic Web: ESWC 2022 Satellite Events - Hersonissos, Crete, Greece, May 29 - June 2, 2022, Proceedings, volume 13384 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 16–20. doi:10.1007/978-3-031-11609-4_3.

[4] R. Iannella, S. Villata, ODRL Information Model 2.2, Recommendation, W3C, 2018. https://www.w3.org/TR/2018/REC-odrl-model-20180215/. Latest version available at https://www.w3.org/TR/odrl-model/.

[5] R. Iannella, M. Steidl, S. Myles, V. Rodríguez-Doncel, ODRL Vocabulary & Expression 2.2, Recommendation, W3C, 2018. https://www.w3.org/TR/2018/REC-odrl-vocab-20180215/. Latest version available at https://www.w3.org/TR/odrl-vocab/.

[6] N. Fornara, V. Rodríguez-Doncel, B. Esteves, S. Steyskal, B. W. Smith, ODRL Formal Semantics, Draft Community Group Report, W3C, 2025. https://w3c.github.io/odrl/formal-semantics/.

[7] S. Steyskal, A. Polleres, Towards formal semantics for ODRL policies, in: N. Bassiliades, G. Gottlob, F. Sadri, A. Paschke, D. Roman (Eds.), Rule Technologies: Foundations, Tools, and Applications - 9th International Symposium, RuleML 2015, Berlin, Germany, August 2-5, 2015, Proceedings, volume 9202 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 360–375. doi:10.1007/978-3-319-21542-6_23.

[8] N. Fornara, M. Colombetti, Operational semantics of an extension of ODRL able to express obligations, in: F. Belardinelli, E. Argente (Eds.), Multi-Agent Systems and Agreement Technologies - EUMAS 2017, and AT 2017, Évry, France, December 14-15, 2017, Revised Selected Papers, volume 10767 of *LNCS*, Springer, 2017, pp. 172–186. doi:10.1007/978-3-030-01713-2_13.

[9] M. D. Vos, S. Kirrane, J. A. Padget, K. Satoh, ODRL policy modelling and compliance checking, in: P. Fodor, M. Montali, D. Calvanese, D. Roman (Eds.), Rules and Reasoning - Third International Joint Conference, RuleML+RR 2019, Bolzano, Italy, September 16-19, 2019, Proceedings, volume 11784 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 36–51. doi:10.1007/978-3-030-31095-0_3.

[10] P. V. Biron, A. Malhotra, XML Schema Part 2: Datatypes Second Edition, Recommendation, W3C, 2004. https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/. Latest version available at https://www.w3.org/TR/xmlschema-2/.
[11] P. Hayes, P. Patel-Schneider, RDF 1.1 Semantics, Recommendation, W3C, 2014. https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/. Latest version available at https://www.w3.org/TR/rdf11-mt/.