# Interoperable Interpretation and Evaluation of ODRL Policies

Wout Slabbinck[1], Julián Andrés Rojas[1], Beatriz Esteves[1], Pieter Colpaert[1], and Ruben Verborgh[1]

IDLab, Dept. Electronics and Information Systems, Ghent University – imec, Belgium
`{firstname.lastname}@ugent.be`

**Abstract.** On the Web, consent banners (cookies) are the prevailing response to legislation such as GDPR for handling protected data. These banners are meant to inform users about how their personal data will be managed by services and third parties. Unfortunately, there is no room for negotiation over these privacy preferences; users either accept the terms to use the Web services or deny them and likely end up with limited or no access. For negotiation to arise, parties must share a common language that is able to describe rights, duties and constraints over digital assets. Given the technical, societal and legal requirements for dealing with privacy preferences, this language must be sufficiently expressive in order to guarantee accuracy. The Open Digital Rights Language (ODRL) standard meets such requirements. However, the lack of a formalism regarding enforcement hinders its adoption. To this end, we propose a systemic approach to interpret and evaluate ODRL Policies, facilitating the creation of interoperable policy engines for enforcing expressive policies. In this paper, we introduce *i)* the Compliance Report Model to denote the result of an ODRL evaluation in an interoperable manner, *ii)* test cases comprised of policies, context and the aforementioned model to ensure correctness of policy engines, *iii)* and the ODRL Evaluator, an implementation that systematically evaluates ODRL policies. We show the expressiveness of our model and the effectiveness of our implementation through an evaluation of the test suite. Addressing the lack of formalisation of ODRL paves the road for negotiation over privacy preferences and establishes the foundations for interoperable policy exchange and evaluations over the Web. Future work includes the further formalisation of the context of policy evaluation and the need for inter-policy strategies for conflict resolution.

**Keywords:** Policy · ODRL · Usage Control

## 1 Introduction

Policy-making is often driven by changes in the real world, reflecting the dynamic relationship between societal needs, technological advancements, and regulatory frameworks. For instance, the rise of digital technologies prompted data protection laws, like the General Data Protection Regulation (GDPR) [2], to address privacy concerns and balance innovation with governance. The emergence of autonomous vehicles has also driven the need to update traffic laws to define liability and safety standards[1] [3]. However, the complex interplay between technology and law is often skewed towards emphasizing how policy-making must restrict technology to safeguard personal data, instead of focusing on how technology can service the enforcement and compliance to such policies. This not only hinders technological innovation, but also perpetuates the current state of distrust on Web-based data exchange practices. This distrust is fed by the current permissionless model on which Web data exchanges are based, where we have to say 'yes' to hundreds of privacy policies through a single interaction in order to have access to a particular piece of content or service on the Web. Such a model eliminates the possibility of having an evolvable relationship between data holders and data consumers, where policies are not universally applicable, but are built throughout the duration of the relationship to service both parties in getting the data and service that they need at the right time.

To achieve such dynamic and scalable systems, two considerations are essential: *i)* since data exchange conditions are not universally defined across parties, these conditions must be articulated through *interoperable* policies, *ii)* these must be sufficiently *expressive* to address technical, societal, and legal frameworks, while allowing the design of systems that can (autonomously) negotiate and agree on the exact conditions for a particular data exchange. Semantic standards are a natural contender for the development of such *policy management systems* or *policy engines*, as they facilitate shared understanding, technical interoperability, and adhere to consensus-driven decision-making processes. Furthermore, policy engines must be deterministic: given a defined set of policies governing the access and usage of a specific piece of data, the output must consistently yield the same set of active rules that must be satisfied to enable the usage of the data. To address such a requirement, policy engines must rely on policy languages that have a standardised interpretation.

As a W3C recommendation, the Open Digital Rights Language (ODRL) stands out as a mature and robust resource, that can facilitate the expression of domain-agnostic, semantic-based rules governing data and service usage. ODRL is already being used, among others, to enable data exchange in common European data spaces [12,21,25] and personal data management scenarios [13,26] alike. Moreover, its extensibility, through profiles, allows it to address specific needs, such as integrating the Data Privacy Vocabulary (DPV) [17] to model legally aligned policies. Nevertheless, ODRL faces a notable limitation: its lack of formalisation is a well-documented issue [5,6,11,18,33], resulting in varying interpretations and inconsistencies in policy enforcement.

In this paper, we propose the ODRL Compliance Report Model to combat the lack of formalisation by expressing the (lack of) compliance of enforcing a given ODRL Policy. Furthermore, we introduce a test suite to standardize

[1] Press release (EU) on new rules to enable driverless vehicles: `https://ec.europa.eu/commission/presscorner/detail/en/ip_22_4312`

the interpretation of ODRL policies, where each test case is comprised of an ODRL Policy in a specific scenario with the expected Compliance Report. Our final contribution is the ODRL Evaluator: an open-source implementation of a policy engine that evaluates ODRL policies and produces a Compliance Report according to the proposed semantics.

The remainder of this article is structured as follows. Section 2 provides an overview of related work. In Section 3, we define the ODRL Compliance Report Model and provide examples for illustration. Section 4 contains the test suite. Section 5 describes the architecture and implementation of the ODRL Evaluator. We evaluate our ODRL Evaluator using the test suite in Section 6. Finally, Section 7 concludes the paper and provides pointers to future research.

## 2 Related Work

### 2.1 Access Control Enforcement on the Web

Initially, the Web was designed solely for information retrieval, without consideration for access control [7]. Basic access authentication, introduced in 1993 and standardized in 1996 as part of HTTP 1.0 [22], was the first move away from the permissionless Web model, allowing servers to restrict access to Web pages. Due to various concerns, including sending credentials with each request and a growing amount of non-interoperable authorization solutions within HTTP servers, a group of organizations proposed OAuth as an open, interoperable authorization specification. Its second version, OAuth 2.0 [15] is currently widely used for enforcing access control on the Web.

These solutions are merely enforcement mechanisms on the Web; they do not dictate the workings of the underlying policy engine. The policy engine follows an algorithm to deterministically calculate whether access should be granted or denied. In the permissionless Web model, that algorithm would always return a "yes," granting access. In the literature, however, there exists a plethora of access control models that utilize specific policy languages to express who can access which resources, enabling engines to handle more expressive policies. Kirrane et al. [19] summarize a multitude of access control models and policy languages. The core of these models lies in using basic access control rules in the form of (subject, resource, access right), with a set of these rules forming a policy. Multiple standardization efforts have taken place to create interoperable policy languages. For instance, there is the eXtensible Access Control Markup Language (XACML) [1] standardisation from OASIS. Other semantic Web standards include the Web Access Control (WAC)[2] and the Access Control Protocol (ACP) specification[3]. To prove their expressiveness and interoperability, formalisms and test suites are employed; a formalism provides consistent mathematical interpretations and proofs, while test suites show correctness through evaluation. For access control,

[2] Web Access Control (WAC): `https://solidproject.org/TR/wac`
[3] Access Control Protocol (ACP): `https://solidproject.org/TR/acp`

test suites exist for XACML[4] and a test harness exists for ACP and WAC (although not as standalone resources)[5].

### 2.2 Usage Control Enforcement on the Web

Unfortunately, access control policies do not allow expressing policies conforming to existing legal frameworks. They lack expressivity for the purpose of access, how it aligns with a legal basis or captures the full use of data. A more expressive language is thus needed, able to capture the full lifecycle of data use: **usage control**. Usage control extends access control through policies containing deontic rules, next to explicit permissions, prohibitions and obligations, and further fine-grained access constraints, which may change over time [23]. From an architectural perspective, to enforce usage control a system needs to support everything related to access control, which becomes a subset also called *preventive usage control* [5]. Furthermore, usage control policies require continuous monitoring of the environment, to ensure access to resources is still permitted in the presence of changes, which is referred to as *detective usage control* [6,24].

For this model, different languages exist. U-XACML that extends XACML to enforce usage control policies [10]. In the Semantic Web, policy languages include ODRL, a W3C Recommendation [31], the SPECIAL Policy Language [8] for expressing consent, business policies and regulatory obligations, and the Data Privacy Vocabulary (DPV) [17] to describe legislative requirements.

**ODRL Policy Engines** The status of W3C Recommendation of ODRL has made it a candidate for usage control enforcement on the Web by initiatives that aim for decentralised data sovereignty such as data spaces[12,21,25] and the Solid initiative [29,26]. While the semantics are standardized, there is a lack of formalisation on how to evaluate ODRL to this date [5,6,11,18,33]. As a result, enforcement systems using ODRL will each have their own interpretation. The following ODRL Policy engines all use the same language, however, they are not interchangeable since part of the context required for evaluating the policy is hard coded within the engines. For **ODRL-PAP**[6], the context is based on custom concepts from the DOME-Marketplace project[7]. Whereas in the **Open Digital Rights Enforcement** (ODRE) Framework from Cimmino et al., [9] time is injected via custom java code and encoded within the policy. The **MOSAICrOWNpolicy** engine [8] relies on the purpose being encoded at the rule level, while ODRL defines purpose as a LeftOperand at the constraint level.

[4] XACML 3.0 Conformance Tests for PDP: `https://github.com/authzforce/core/blob/develop/pdp-testutils/src/test/resources/conformance/xacml-3.0-from-2.0-ct/README.md`

[5] `https://github.com/solid-contrib/conformance-test-harness/tree/main`

[6] ODRL-PAP: `https://github.com/wistefan/odrl-pap`

[7] DOME-Marketplace project: `https://dome-marketplace.eu/`

[8] MOSAICrOWNpolicy policy engine: `https://github.com/mosaicrown/policy-engine`

Hosseinzadeh et al. [16] introduced a policy engine, **MYDATA Control Technologies**, that transforms ODRL policies to the custom XML-based MYDATA policy language through mapping rules. The Prometheus-X ODRL manager[9], relies on fixed structural elements of JSON-based data and does not handle syntax-agnostic RDF nature of ODRL policies in general.

To overcome the lack of interoperable ODRL Policy engines, Steyskal and Polleres argued for a semantic formalism over ODRL to enable rule-based reasoning over its policies [27]. To this end, an ODRL Community Group[10] was formed with the goal of creating an ODRL Evaluator and formalise its expected behaviour. To the best of our knowledge, there exists no implementation of an ODRL Policy engine following formal semantics nor a test suite to guide other implementations.

## 3 Compliance Report Model

In ODRL, an ODRL Evaluator is defined as follows *"A system that determines whether the Rules of an ODRL Policy expression have meet their intended action performance"* [31]. However, a problem arises when trying to build an ODRL Evaluator; more precisely: how is *action performance* modelled? ODRL defines policies consisting of rules representing the deontic concepts such as permission, prohibition or obligation, and each rule can have several constraints. Furthermore, ODRL specifies for each class of rule when the action is allowed or not based on whether all constraints and refinements are satisfied, meaning the comparison returns a match. What ODRL does not define is how to obtain the values for those comparisons; for instance, real-world information like *time*. From now on, such information is referred to as the **State of the World** (SotW).

**Definition 1 (State of the World).** *A set of knowledge representing real-world information aiding the evaluation of ODRL Policies.*

In addition, ODRL neither defines how to model access requests, which are necessary to match other obligatory rule properties, such as the party, action and target of the rule. Consequently, the existing policy engines that evaluate ODRL policies are not interoperable due to the lack of formal enforcement semantics.

To this end, we have created the **Compliance Report Model**[11], a specialized vocabulary designed to articulate the degree of compliance or non-compliance with ODRL policies and the State of the World. It serves primarily as an output of ODRL Evaluation, explaining past executed actions. The model is versatile and applicable for both *preventive* usage control—where it determines whether an access request should be granted — and *detective* usage control—to verify that actions align with the deontic meaning of the given rules. The remaining

[9] Prometheus-X ODRL manager: `https://github.com/Prometheus-X-association/odrl-manager`

[10] ODRL Formal Semantics specification: `https://w3c.github.io/odrl/formal-semantics/`

[11] ODRL Compliance Report Model: `https://w3id.org/force/compliance-report`

subsections elaborate on *i)* the definitions of the vocabulary; *ii)* how the report elaborates whether an action can be executed in a preventive usage control setting, and; *iii)* whether an action was allowed to be executed in a detective usage control setting.

## 3.1 Compliance Report Model Definitions

The *Compliance Report Model* provides a structured framework for evaluating ODRL policies. An ODRL Policy comprises one or multiple rules, each containing properties and potential constraints. Satisfaction of these properties and constraints affects the performance (or lack thereof) of an action based on the deontic concept, indicating whether an action may, must not/should not, or must be performed. Therefore, there are different levels of *reports*. Furthermore, we have *states* which are tied to reports that capture the state of evaluation. The top-level report is the **Policy Report**.

**Definition 2 (Policy Report).** *The compliance result of an ODRL Evaluation over an `odrl:Policy`, the State of the World, and, in the case of an access control scenario, an `odrl:Request`.*

Determining the compliance of an ODRL Policy relies not only on its rules, but also on real-world information. This includes the *time* when an evaluation was made, the roles of a party (group memberships), and the groupings of a target resource (resource memberships in a collection). As this information is dynamic (it changes over time) the evaluation must be based on a snapshot at a given point in time that considers all information known at that moment. In other words, a Policy Report captures the compliance to an ODRL Policy over a snapshot at a given moment in time, distinguished by a timestamp using the `dct:created` property. An ODRL Policy consists of at least one rule, and to denote compliance for each rule, the Policy Report includes a **Rule Report**.

**Definition 3 (Rule Report).** *The compliance result of an ODRL Evaluation over an `odrl:Rule` and, in the case of preventive usage control, an `odrl:Request`.*

Rule Reports have three subclasses that encapsulate the rule's deontic concept: *Permission Report*, *Prohibition Report*, and *Duty Report*, which relate to the corresponding specific ODRL rules `odrl:Permission`, `odrl:Prohibition` and `odrl:Duty`. To denote compliance to conditions of a rule, **Premise Reports** are used, which are linked from *Rule Reports.*

**Definition 4 (Premise Report).** *A generic compliance report for any kind of constraint that must be satisfied for a rule to become active.*

There are four specific *Premise Reports*: *Constraint Report*, *Party Report*, *Target Report*, and *Action Report.* They relate correspondingly to satisfaction of ODRL Constraints and, in the case of preventive usage control, the matching or inclusion of target, party and action of the request and policy.

Furthermore, a *Rule Report* encompasses four states which are based on the ODRL formalization work of Fornara and Colombetti [14]: *Activation State*, *Attempted State*, *Performance State*, and *Deontic State*. The *Activation State* indicates whether an `odrl:Rule` is active or inactive. The meaning of *active* depends on the deontic concept of the rule: for permission, the action may be performed; for prohibition, it may not; for duty, it must be performed. The *Attempted State* denotes whether there was a request that has been attempted to perform the action of a rule. The *Performance State* reflects the knowledge of whether the action of a `odrl:Rule` has been performed. The possible Performance States can be **Unknown**, indicating an explicit lack of knowledge regarding the action performance status; **Performed**, the action was performed; or **Unperformed**, meaning that the action was not performed and can no longer be performed. And finally, the *Deontic State*, which is a function over the deontic concept, Activation State and Performance State of the `odrl:Rule` entailing three possible states. First, there is *Violated*, which is when the combination of Activation State and Performance State contradicts the deontic concept of the rule. Next, when that combination complies with the deontic concept, the Deontic State becomes *Fulfilled*. Lastly, for the cases where no clear conclusion can be made whether the Deontic State is Violated or Fulfilled, there is the *NonSet* state.

The Activation State for a given deontic concept dictates whether the action may be performed or not and the Deontic State indicates whether those instructions have been followed, thereby enabling the swift detection of malicious behaviour. For a *permission*, the action can only be executed when all premises are satisfied and when the duties are not violated. So if this was not the case (*inactive*) and the action has been *performed*, there is a contradiction resulting in a *violated* state of the rule. Analogously, a *prohibition* that is *active* means that the action must not be performed. There is again a contradiction when the rule's action is *performed*, resulting in the Deontic State being *violated* again. However, if it is known that the action will never be performed (*unperformed*), compliance to the deontic concept is guaranteed and as a consequence, the Deontic State is *fulfilled*. The remainder of the Deontic State calculations follow similar reasoning and these states are presented in Table 1.

## 3.2 Preventive Usage Control

In this scenario, the Policy Report is the result of evaluating an ODRL Policy, the SotW, and an access request and it determines whether an action may or may not be performed. In the case it may not be performed, the report explains the exact reason by linking to the premise that has not been fulfilled. For the remainder of the paper, we will re-use the `odrl:Request` policy with a Permission rule to represent an access request and compliance report to represent the Policy Report along with all its Rule and Premise Reports.

To elaborate the compliance report, consider the following example. Bob hosts a resource X, with identifier `ex:x`, on a server equipped with an Authorization component that enforces ODRL Policies to prevent unauthorized access. Alice, an acquaintance of Bob, seeks read access to resource X for the purpose of reading its

| **Deontic Concept** | **Activation State** | **Performance State** | **Deontic State** |
|---|---|---|---|
| **Permission** | Active | Performed | *Fulfilled* |
| | | Unperformed | *Not-set* |
| | | Unknown | *Not-set* |
| | Inactive | Performed | *Violated* |
| | | Unperformed | *Fulfilled* |
| | | Unknown | *Not-set* |
| **Prohibition** | Active | Performed | *Violated* |
| | | Unperformed | *Fulfilled* |
| | | Unknown | *Not-set* |
| | Inactive | Performed | *Not-set* |
| | | Unperformed | *Not-set* |
| | | Unknown | *Not-set* |
| **Duty** | Active | Performed | *Fulfilled* |
| | | Unperformed | *Violated* |
| | | Unknown | *Not-set* |
| | Inactive | Performed | *Not-set* |
| | | Unperformed | *Not-set* |
| | | Unknown | *Not-set* |

**Table 1:** The Deontic States in function of the Deontic Concept, Activation State and Performance State.

contents, as depicted in Listing 2. Anticipating this scenario, Bob has configured the Authorization component with the ODRL Policy outlined in Listing 1, which explicitly grants Alice permission to read the resource.

Alice's request arrives at the system at time $t_1$ (2024-02-12T11:20:10), as represented by the SotW in Listing 3. The policy engine within the Authorization component evaluates the policy, SotW, and request, producing an ODRL Compliance Policy Report, shown in Listing 4. The compliance report elaborates that `ex:alice` is explicitly allowed to perform the action `odrl:read` on resource `ex:x`, as the rule is active. This rule is active because all its premises are satisfied: the action, resource (target), and party match. Furthermore, the compliance report provides provenance through `dct:created`, indicating the request was received at time $t_1$.

## 3.3 Detective Usage Control

In this scenario, the Policy Report results from evaluating an ODRL Policy and the SotW. It determines whether an action was allowed to be performed, which is made clear through the Deontic State.

To elucidate such a compliance report, we continue with the example from Preventive Usage Control (3.2). One second after Alice requested to read resource X ($t_2$), she performed the action. Bob wants to verify that all actions on his server conform to the policies. Using the new compliance report created by the policy engine, depicted in Listing 5, he can confirm that the read action on resource X has indeed been performed and that the Deontic State was fulfilled.

```
ex:odrlPolicy a odrl:Set;
  odrl:uid ex:odrlPolicy;
  odrl:permission ex:policyRule.

ex:policyRule a odrl:Permission;
  odrl:action odrl:read;
  odrl:target ex:x;
  odrl:assignee ex:alice.
```

**Listing 1:** ODRL Policy with a permission Rule that states: **permission** for **alice** to **read x**.

```
ex:odrlRequest a odrl:Request;
  odrl:uid ex:odrlRequest;
  odrl:permission ex:requestRule.

ex:requestRule a odrl:Permission;
  odrl:assignee ex:alice;
  odrl:action odrl:read;
  odrl:target ex:x.
```

**Listing 2:** Access request represented as an ODRL Request with a Permission rule that states the following: may **alice read** resource **x**?

```
temp:currentTime dct:issued "2024-02-12T11:20:10Z"^^xsd:dateTime.
```

**Listing 3:** State of the World representing the current time.

```
ex:policyReport1 a report:PolicyReport;
  dct:created "2024-02-12T11:20:10Z"^^xsd:dateTime ;
  report:policy ex:odrlPolicy ;
  report:policyRequest ex:odrlRequest ;
  report:ruleReport ex:permissionReport1 .

ex:permissionReport1 a report:PermissionReport ;
  report:rule ex:policyRule ;
  report:ruleRequest ex:requestRule ;
  report:activationState report:Active ;
  report:attemptState report:Attempted ;
  report:premiseReport ex:partyReport, ex:targetReport, ex:actionReport.

ex:partyReport a report:PartyReport;
  report:satisfactionState report:Satisfied .

ex:targetReport a report:TargetReport;
  report:satisfactionState report:Satisfied .

ex:actionReport a report:ActionReport;
  report:satisfactionState report:Satisfied .
```

**Listing 4:** Compliance Report of an ODRL Policy (1), an access request (2) given the SotW (3).

```
ex:policyReport2 a report:PolicyReport;
  dct:created "2024-02-12T11:20:11Z"^^xsd:dateTime ;
  report:policy ex:odrlPolicy ;
  report:policyRequest ex:odrlRequest ;
  report:ruleReport ex:permissionReport2 .

ex:permissionReport2 a report:PermissionReport ;
  report:rule ex:policyRule ;
  report:ruleRequest ex:requestRule ;
  report:activationState report:Active ;
  report:attemptState report: ;
  report:performanceState report:Performed ;
  report:deonticState report:Fulfilled ;
  report:premiseReport ex:partyReport, ex:targetReport, ex:actionReport.
...
```

**Listing 5:** Compliance Report of an ODRL Policy (1) when access request (2) is performed at $t_2$, a later time than the SotW (3) (Premise Reports are omitted for brevity).

## 4 Test Suite

In this section, we describe the test suite designed to ensure ODRL Evaluators are **consistent**, **verified**, and **reliable**[12]. Ensuring different ODRL implementations interpret policies consistently is crucial, and a test suite helps standardize outputs, reducing discrepancies and ambiguities. Additionally, it verifies the correctness of evaluators across various scenarios. Furthermore, continuously passing test suites increases confidence in the reliability of an ODRL Evaluator. Figure 1 illustrates the architecture of the test suite, outlining the high-level flow and components involved in the evaluation process.

The test suite encompasses various test cases, each containing an ODRL Policy, the State of the World, and a Policy Compliance Report, as illustrated in Listing 6. Additionally, preventive usage control test cases include an ODRL request. The Compliance Report in the test case provides the expected Policy Report that an ODRL Evaluator must produce given the test inputs. This variety ensures that all intricacies and edge cases of ODRL Policies are tested in isolation.

The test cases are executed using black-box testing, meaning the test suite is unaware of the internal behaviours of any ODRL evaluator. To compare the output of an ODRL Evaluator with the expected Policy Report, the structure and states are compared. The output must match the expected compliance report in terms of the number and relationships of rule and Premise Reports, and reference the same ODRL Policies, Rules and Constraints. Additionally, each Premise Report's satisfaction state, and each Rule Report's Attempted, Activation, Performance, and Deontic States, must be identical.

[12] ODRL Test suite Repository: `https://zenodo.org/records/14290518`

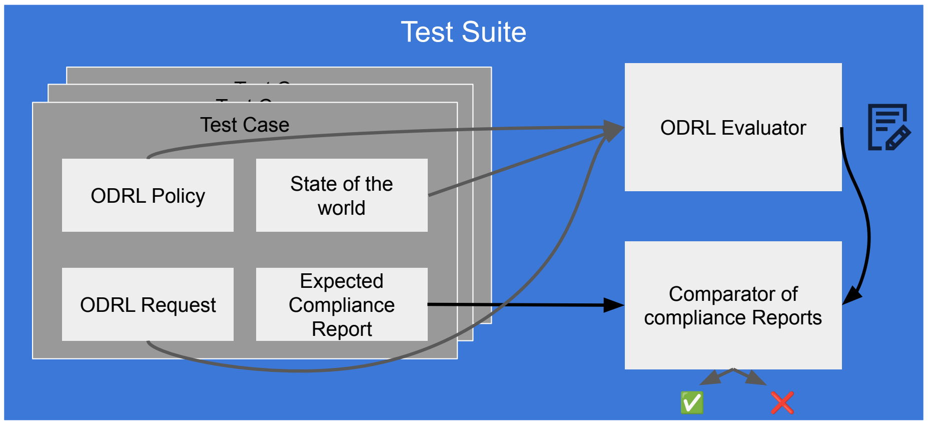

**Fig. 1:** Test Suite for ODRL Evaluators showing two phases: (i) ODRL Evaluation over an ODRL Policy, the SotW and (optionally) an ODRL Request *(grey arrows)* (ii), and comparing the result of the generated to an expected Policy Report *(black arrows)*.

```
ex:testCase1 a ex:TestCase;
  dct:title "Title of the test case.";
  ex:policy ex:odrlPolicy ;
  ex:request ex:odrlRequest ;
  ex:sotw <StateOfTheWorldIdentifier> ;
  ex:expectedReport ex:policyReport1 .
```

**Listing 6:** Test case expecting that an ODRL evaluation of the ODRL Policy in Listing 1, access request in Listing 2, and at SotW in Listing 3 corresponds to the compliance report in Listing 4.

## 5 ODRL Evaluator

In this section, we present our ODRL Evaluator, a software component to systematically generate the compliance reports from ODRL Policies and the State of the World. It consists of two key components: a **validation component**, which performs preprocessing steps to clean and validate for syntactic and semantic correctness, and an **evaluation component** where ODRL Rules are interpreted to generate the ODRL Compliance Report. Figure 2 illustrates these components, which are further elaborated in the next paragraphs. Note that an ODRL Evaluator does not decide whether an action will be *performed*, and thus does not produce performance and Deontic States; rather, it provides a decision on potential action performance via the Activation State.

The validation component executes several checks. First, the **Ensure Triples** step verifies that the ODRL Evaluator only accepts RDFJS Datasets[13]. Next, to ensure at least one ODRL Policy is present **Cardinality Check** is executed.

[13] RDFJS Dataset specification: `https://rdf.js.org/dataset-spec/`

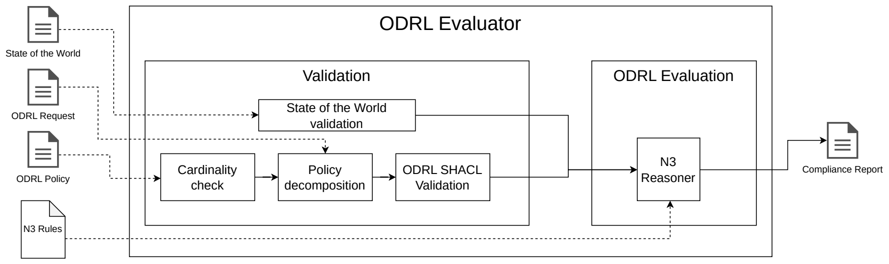

**Fig. 2:** Architecture of the ODRL Evaluator illustrating the flow: (i) policies and SotW flow to the validation steps, (ii) validation occurs in the **Validation component**, and (iii) evaluation in the **ODRL Evaluation component** using validated inputs and N3 rules to produce a Compliance Report.

Following this, the **Policy Decomposition** step guarantees that all the policies and requests are atomic[14]. Additionally, the **ODRL SHACL Validation** confirms that the ODRL Policy and (optional) ODRL Request conform to a valid SHACL [20] shape, adhering to the ODRL Information Model standard. Finally, **State of the World Validation** verifies that the State of the World contains the minimum required properties for the ODRL Evaluator to function, such as the current time or any ODRL party or asset collection.

Once the input is verified, the actual ODRL Evaluation can be performed, during which compliance reports are deduced through reasoning. For this reasoning, declarative rules are written in Notation3 [28] and executed by the EYE reasoner in TypeScript code[15]. Before explaining how the rules work together to produce a compliance report, we will first elaborate on the set of rules and their functions:

- **Inference rules**: These contain three types of inferences over the ODRL model: (i) higher-level concept inferences via `rdfs:Class` and `rdfs:ClassOf`, (ii) SKOS[16] terms inferences such as `skos:exactMatch`, and (iii) action transitivity inferences through the `odrl:includedIn` property.
- **Structural rules**: These conclude the Policy, Rule, and Premise Reports with corresponding relationships based on the provided ODRL policies of class `Set`, `Offer`, and `Agreement`, as well as the time of the SotW, are concluded.
- **Satisfaction rules**: These infer the satisfaction states of Premise Reports. **Inference-based pattern matching**, including transitivity and Party/Asset Collections, is performed for access requests to derive the satisfaction state for Target, Party, and Action Reports. There are two types of rules for the Constraint Report: **instantiated** and **compound rules**. The former entails

[14] *Atomic* level Policy described in the composition section of ODRL IM 2.2: `https://www.w3.org/TR/odrl-model/#composition`
[15] through EYE JS [32] by default, but normal EYE [30] can be used as well
[16] Simple Knowledge Organization System (SKOS): `https://www.w3.org/TR/skos-reference/`

evaluation based on an operator and the right and left operands, whereas the latter is achieved through rules for `odrl:LeftOperand` instantiation by deriving facts from the SotW. The latter deduces the satisfaction state from multiple Premise Reports based on the logical `odrl:operand`.

- **Activation rules**: These conclude the Activation State in the Rule Reports based on the satisfaction states of their related Premise Reports. For permission rules, this Activation State is also dependent on the Deontic State of the related Duty Deport for its duties.
- **Deontic rules**: These deduce the Deontic State of the rule action based on the performance, satisfaction state, and the deontic concept.

Evaluating ODRL policies is further split into two separate reasoning configurations. The first configuration generates new compliance reports for preventive usage control, while the second deduces Deontic States based on existing compliance reports.

The preventive usage control configuration consists of multiple runs, each using a different rule set. Multiple runs are necessary to gradually build up the compliance reports due to the set of rules are forward-chaining, which entails the reasoning engine iteratively generating new facts until saturation[17]. Inference rules are employed in all runs. During the first run, Structural rules generate stateless Policy, Rule, and Premise Reports. The second run calculates all satisfaction states of the premises using Satisfaction rules. Once these states are known, the Activation rules calculate the Activation State of the Rule Reports.

The detective usage control configuration involves a single run using Deontic rules to calculate the Deontic State of the Rule Reports.

Combining the two configurations into a third is theoretically possible by first executing the preventive and then the detective configuration. However, as stated earlier, this approach is not fruitful since action performance is known after generating the preventive usage control compliance report.

The ODRL Evaluator is an open-source TypeScript package[18] and can be configured to run in browsers using the EYE JS reasoner.

## 6 Evaluation and Discussion

In this section, we demonstrate that the Compliance Report Model captures the expressiveness needed for complex usage control scenarios by validating it against our test suite of 68 test cases. Furthermore, as the test suite is designed to validate the implementation of ODRL evaluators, we show the capabilities of our presented ODRL evaluator. Lastly, we discuss the limitations of our work and offer potential solutions.

[17] If all reasoning steps were attempted at once, conflicts could arise, such as the Activation State being both active and inactive.

[18] npm package ODRL Evaluator: `https://www.npmjs.com/package/odrl-evaluator`

| Test cases | Constant | Access Control | Logical Constraints | Constraints | Group | Duties |
|---|---|---|---|---|---|---|
| 1-6 | ✓ | | | | | |
| 7-29 | | ✓ | | | | |
| 30-47 | | ✓ | | ✓ | | |
| 48-50, 62-64 | | ✓ | ✓ | ✓ | | |
| 51-58 | | ✓ | | | ✓ | |
| 59-61 | | ✓ | | | | ✓ |
| 65-68 | | ✓ | ✓ | ✓ | ✓ | ✓ |

**Table 2:** Test cases and the respective expressitivity categories they belong to.

## 6.1 Expressitivity

Table 2 shows the expressiveness of the test cases of the test suite, grouped into six categories depending on the expressiveness of the policy and the respective expected Policy Report.

The **constant** category includes test cases where the expected Policy Report is only dependent on the policy, irrespective of the request and the SotW. The expected Rule Report will indicate either an access grant through an active permission or an access denial through an active prohibition. The **access control** category entails test cases where the Activation States in the expected Rule Report are dependent both on the policy and request. Specifically, it is dependent on the matching of party, target, and action (including hierarchical inferences defined in ODRL, such as `odrl:read` being included in `odrl:use`). In the **logical constraints** category, test cases include policies with a set of constraints. The Activation State of the expected Rule Report depends on one or multiple logical operations over those constraints, which include conjunction, disjunction and uniqueness quantification. The **constraints** category consists of test cases where the policies have constraints where the left operand needs to be provided through the SotW. Currently, only temporal constraints are included, since we solely demonstrated how to retrieve *time* from the SotW, though generalization is straightforward. The test cases from the **group** category contain policies with rules that restrict access to either a party group and/or where the target of the rule is a group of resources, where the group memberships are supplied through the SotW. Finally, the **duties** category contains test cases where there are policies with a permission rule containing duties. As a result, Activation States depend on the Deontic States of the Duty Reports, which are supplied through the SoTW.

## 6.2 ODRL Evaluator Coverage

While the Compliance Report Model provides generic terminology to show compliance to ODRL policies, the ODRL evaluator still lacks the capabilities to evaluate all policies crafted with the ODRL Core Vocabulary. For the subclasses of the ODRL Left Operand, we implemented `odrl:dateTime` by using a triple representing the time in the SotW. Although other subclasses are not yet implemented, a similar approach could be easily applied to enable their evaluation. Additionally, the ODRL Evaluator does not assess set-based operators due to the absence of examples, semantics, and formalisation in the ODRL Information Model. A lack of semantics and formalisation also explains why the ODRL Evaluator cannot handle ODRL Profiles. Finally, the ODRL Evaluator does not yet incorporate the Deontic State of Duty Reports for determining Activation State of permissions. All other things of the ODRL Core Vocabulary has been implemented[19], thereby handling 61 out of 68 test cases .

### 6.3 Discussion

The Policy Compliance Report captures the evaluation of a single policy. When multiple policies with contradicting rules are provided — such as an access request resulting in both a Permission and Prohibition Report that simultaneously grant and deny the action — a conflicting answer is received, necessitating an intervention for enforcement. This issue is recurring in usage control research, where solutions often point to implementing a conflict resolution strategy [4,27].

We focused on preventive usage control policy test cases as detective policy evaluation[20] only considers three variables to generate the Deontic State (Table 1).

## 7 Conclusion

In this paper, we argue for the need for *interoperable* policy engines that can handle *expressive* policies. We identify the ODRL standard as the prime policy language candidate. Due to its lack of semantics for enforcement, we propose the interoperable Compliance Report Model to represent the output of ODRL Policy evaluation. Additionally, we emphasize the necessity explicitly incorporating real-world information as input to ODRL Evaluators to ensure consistent evaluation across ODRL Policy engines. Furthermore, we introduce the first test suite for ODRL Evaluators, enabling the assessment of their consistency and reliability. Finally, we present a deterministic, open-source ODRL Evaluator with clear transparency regarding its level of ODRL support.

The research presented in this paper identifies several avenues for future work to support the widespread adoption of interoperable policy engines. This includes further standardization and formalization of ODRL evaluations, potentially through the development of mathematical proofs to establish a rigorous basis for policy evaluation and enforcement. These efforts should encompass conflict resolution strategies between policies and methods for incorporating real-world information. Once standardized, attention can then shift to optimizing performance to handle complex policies over large datasets effectively.

[19] ODRL Evaluator Support: `https://github.com/SolidLabResearch/ODRL-Evaluator/blob/main/ODRL-Support.md`

[20] N3 Rule for Deontic State calculation: `https://github.com/SolidLabResearch/ODRL-Evaluator/blob/main/src/rules/detective/deontic-states.n3`

**Acknowledgments**

This research was funded by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10) and by the imec.icon project PACSOI (HBC.2023.0752), which was co-financed by imec and VLAIO and brings together the following partners: FAQIR Foundation, FAQIR Institute, MoveUP, Byteflies, AContrario, and Ghent University – IDLab.

## References

1. eXtensible Access Control Markup Language (XACML) v3.0 (2013), `https://www.oasis-open.org/standard/xacmlv3-0/`
2. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance) (Apr 2016), `http://data.europa.eu/eli/reg/2016/679/oj/eng`, legislative Body: EP, CONSIL
3. Regulation - 2019/2144 - EN - EUR-Lex (Nov 2019), `https://eur-lex.europa.eu/eli/reg/2019/2144/oj/eng`, doc ID: 32019R2144 Doc Sector: 3 Doc Title: Regulation (EU) 2019/2144 of the European Parliament and of the Council of 27 November 2019 on type-approval requirements for motor vehicles and their trailers, and systems, components and separate technical units intended for such vehicles, as regards their general safety and the protection of vehicle occupants and vulnerable road users, amending Regulation (EU) 2018/858 of the European Parliament and of the Council and repealing Regulations (EC) No 78/2009, (EC) No 79/2009 and (EC) No 661/2009 of the European Parliament and of the Council and Commission Regulations (EC) No 631/2009, (EU) No 406/2010, (EU) No 672/2010, (EU) No 1003/2010, (EU) No 1005/2010, (EU) No 1008/2010, (EU) No 1009/2010, (EU) No 19/2011, (EU) No 109/2011, (EU) No 458/2011, (EU) No 65/2012, (EU) No 130/2012, (EU) No 347/2012, (EU) No 351/2012, (EU) No 1230/2012 and (EU) 2015/166 (Text with EEA relevance) Doc Type: R Usr_lan: en
4. Akaichi, I., Flouris, G., Fundulaki, I., Kirrane, S.: GUCON: A Generic Graph Pattern Based Policy Framework for Usage Control Enforcement. In: Fensel, A., Ozaki, A., Roman, D., Soylu, A. (eds.) Rules and Reasoning, vol. 14244, pp. 34–53. Springer Nature Switzerland, Cham (2023). `https://doi.org/10.1007/978-3-031-45072-3_3`, `https://link.springer.com/10.1007/978-3-031-45072-3_3`, series Title: Lecture Notes in Computer Science
5. Akaichi, I., Kirrane, S.: Usage Control Specification, Enforcement, and Robustness: A Survey (Mar 2022). `https://doi.org/10.48550/arXiv.2203.04800`, `http://arxiv.org/abs/2203.04800`, arXiv:2203.04800 [cs]
6. Akaichi, I., Slabbinck, W., Rojas, J.A., Gheluwe, C.V., Bozzi, G., Colpaert, P., Verborgh, R., Kirrane, S.: Interoperable and Continuous Usage Control Enforcement in Dataspaces. In: Theissen-Lipp, J., Colpaert, P., Sowe, S.K., Curry, E., Decker, S. (eds.) Proceedings of the Second International Workshop on Semantics in Dataspaces (SDS 2024). CEUR Workshop Proceedings, vol. 3705. CEUR, Hersonissos, Greece (May 2024), `https://ceur-ws.org/Vol-3705/#paper10`, iSSN: 1613-0073
7. Berners-Lee, T., Cailliau, R., Groff, J.F., Pollermann, B.: World-Wide Web: the information universe. Internet Research **2**(1), 52–58 (1992), `https://www.emerald.com/insight/content/doi/10.1108/eb047254/full/html`, publisher: MCB UP Ltd

8. Bonatti, P.A., Kirrane, S., Petrova, I.M., Sauro, L.: Machine Understandable Policies and GDPR Compliance Checking. KI - Künstliche Intelligenz **34**(3), 303–315 (Sep 2020). https://doi.org/10.1007/s13218-020-00677-4, https://doi.org/10.1007/s13218-020-00677-4
9. Cimmino, A., Cano-Benito, J., García-Castro, R.: Open Digital Rights Enforcement framework (ODRE): From descriptive to enforceable policies. Comput. Secur. **150**(C) (Feb 2025). https://doi.org/10.1016/j.cose.2024.104282, https://doi.org/10.1016/j.cose.2024.104282
10. Colombo, M., Lazouski, A., Martinelli, F., Mori, P.: A Proposal on Enhancing XACML with Continuous Usage Control Features. In: Desprez, F., Getov, V., Priol, T., Yahyapour, R. (eds.) Grids, P2P and Services Computing, pp. 133–146. Springer US, Boston, MA (2010). https://doi.org/10.1007/978-1-4419-6794-7_11, https://link.springer.com/10.1007/978-1-4419-6794-7_11
11. De Vos, M., Kirrane, S., Padget, J., Satoh, K.: ODRL policy modelling and compliance checking. In: Rules and Reasoning: Third International Joint Conference, RuleML+ RR 2019, Bolzano, Italy, September 16–19, 2019, Proceedings 3. pp. 36–51. Springer (2019)
12. Drees, H., Kubitza, D.O., Lipp, J., Pretzsch, S., Langdon, C.S.: Mobility data space–first implementation and business opportunities. In: ITS World Congress (2021), https://www.researchgate.net/profile/Johannes-Theissen-Lipp/publication/351519610_Mobility_Data_Space_-_First_Implementation_and_Business_Opportunities/links/610101882bf3553b29174ee6/Mobility-Data-Space-First-Implementation-and-Business-Opportunities.pdf
13. Esteves, B., Rodríguez-Doncel, V., Pandit, H.J., Mondada, N., McBennett, P.: Using the ODRL profile for access control for solid pod resource governance. In: European Semantic Web Conference. pp. 16–20. Springer (2022)
14. Fornara, N., Colombetti, M.: Using semantic web technologies and production rules for reasoning on obligations, permissions, and prohibitions. Ai Communications **32**(4), 319–334 (2019), https://content.iospress.com/articles/ai-communications/aic190617, publisher: IOS Press
15. Hardt, D.: The OAuth 2.0 Authorization Framework. Request for Comments RFC 6749, Internet Engineering Task Force (Oct 2012). https://doi.org/10.17487/RFC6749, https://datatracker.ietf.org/doc/rfc6749, num Pages: 76
16. Hosseinzadeh, A., Eitel, A., Jung, C.: A Systematic Approach toward Extracting Technically Enforceable Policies from Data Usage Control Requirements:. In: Proceedings of the 6th International Conference on Information Systems Security and Privacy. pp. 397–405. SCITEPRESS - Science and Technology Publications, Valletta, Malta (2020). https://doi.org/10.5220/0008936003970405, http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0008936003970405
17. J. Pandit, H., Esteves, B., P. Krog, G., Ryan, P., Golpayegani, D., Flake, J.: Data Privacy Vocabulary (DPV) – Version 2.0. In: Demartini, G., Hose, K., Acosta, M., Palmonari, M., Cheng, G., Skaf-Molli, H., Ferranti, N., Hernández, D., Hogan, A. (eds.) The Semantic Web – ISWC 2024, vol. 15233, pp. 171–193. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-77847-6_10, https://link.springer.com/10.1007/978-3-031-77847-6_10, series Title: Lecture Notes in Computer Science
18. Kebede, M.G., Sileno, G., Van Engers, T.: A critical reflection on ODRL. In: International Workshop on AI Approaches to the Complexity of Legal Systems. pp. 48–61. Springer (2018)

19. Kirrane, S., Mileo, A., Decker, S.: Access Control and the Resource Description Framework: A Survey. Semant. Web **8**(2), 311–352 (Jan 2017). `https://doi.org/10.3233/SW-160236`, `https://doi.org/10.3233/SW-160236`, place: NLD Publisher: IOS Press
20. Knublauch, H., Kontokostas, D.: Shapes constraint language (SHACL). W3C Recommendation **11**(8), 1 (2017), `https://www.w3.org/TR/shacl/`
21. Nagel, L., Lycklama, D.: Design Principles for Data Spaces - Position Paper. Tech. rep., Zenodo (Jul 2021). `https://doi.org/10.5281/zenodo.5105744`, `https://zenodo.org/records/5105744`
22. Nielsen, H., Fielding, R.T., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.0. Request for Comments RFC 1945, Internet Engineering Task Force (May 1996). `https://doi.org/10.17487/RFC1945`, `https://datatracker.ietf.org/doc/rfc1945`, num Pages: 60
23. Park, J., Sandhu, R.: The UCONABC usage control model. ACM Transactions on Information and System Security **7**(1), 128–174 (Feb 2004). `https://doi.org/10.1145/984334.984339`, `https://doi.org/10.1145/984334.984339`
24. Pretschner, A., Hilty, M., Schütz, F., Schaefer, C., Walter, T.: Usage Control Enforcement: Present and Future. IEEE Security & Privacy Magazine **6**(4), 44–53 (Jul 2008). `https://doi.org/10.1109/MSP.2008.101`, `http://ieeexplore.ieee.org/document/4588229/`
25. Siska, V., Karagiannis, V., Drobics, M.: Building a Dataspace: Technical Overview (2023), `https://www.gaia-x.at/wp-content/uploads/2023/04/WhitepaperGaiaX.pdf`
26. Slabbinck, W., Rojas Meléndez, J.A., Esteves, B., Verborgh, R., Colpaert, P.: Enforcing Usage Control Policies in Solid Using a Rule-Based Software Agent. In: Proceedings of the Second Solid Symposium (May 2024)
27. Steyskal, S., Polleres, A.: Towards Formal Semantics for ODRL Policies. In: Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., Roman, D. (eds.) Rule Technologies: Foundations, Tools, and Applications. pp. 360–375. Springer International Publishing, Cham (2015). `https://doi.org/10.1007/978-3-319-21542-6_23`
28. Tim Berners-Lee: Notation 3 Logic (1998), `https://www.w3.org/DesignIssues/Notation3.html`
29. Verborgh, R.: Re-Decentralizing the Web, For Good This Time. In: Seneviratne, O., Hendler, J. (eds.) Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web, pp. 215–230. ACM (Sep 2023). `https://doi.org/10.1145/3591366.3591385`, `https://ruben.verborgh.org/articles/redecentralizing-the-web/`
30. Verborgh, R., De Roo, J.: Drawing Conclusions from Linked Data on the Web: The EYE Reasoner. IEEE Software **32**(3), 23–27 (May 2015). `https://doi.org/10.1109/MS.2015.63`, `http://ieeexplore.ieee.org/document/7093047/`
31. W3C Working Group: The Open Digital Rights Language (ODRL) (2018), `https://www.w3.org/TR/odrl-model/`
32. Wright, J., Roo, J.D., Smessaert, I.: EYE JS: A client-side reasoning engine supporting Notation3 and RDF Surfaces. In: Etcheverry, L., Garcia, V.L., Osborne, F., Pernisch, R. (eds.) Proceedings of the ISWC 2024 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice. CEUR Workshop Proceedings, vol. 3828. CEUR, Maryland, USA (Nov 2024), `https://ceur-ws.org/Vol-3828/#ISWC2024_paper_8`, iSSN: 1613-0073
33. Zhao, R., Zhao, J.: Perennial Semantic Data Terms of Use for Decentralized Web. In: Proceedings of the ACM Web Conference 2024. pp. 2238–2249. ACM,

Singapore Singapore (May 2024). https://doi.org/10.1145/3589334.3645631, https://dl.acm.org/doi/10.1145/3589334.3645631