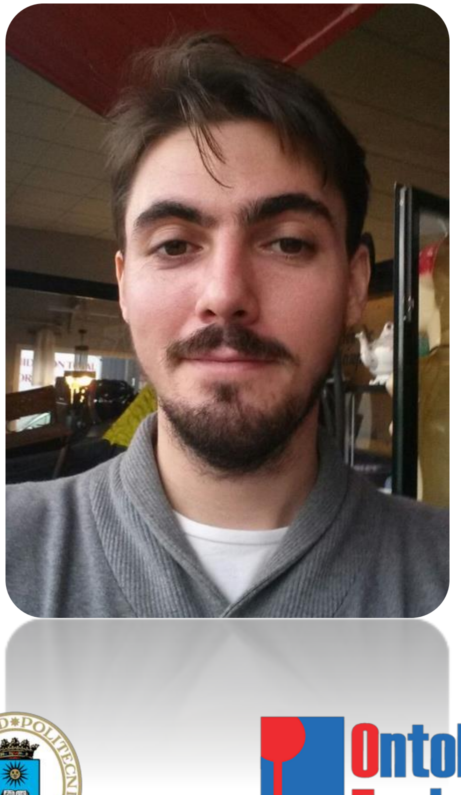
Open Digital Rights Enforcement Framework (ODRE): from descriptive to enforceable policies

Andrea Cimmino, Juan Cano Benito and Raúl García Castro
Ontology Engineering Group
Universidad Politécnica de Madrid, Spain

andreajesus.cimmino@upm.es
@acimmino

A short presentation



Permanent Professor

Senior Researcher

Standardization

Web of Things (WoT) Discovery
W3C Recommendation 05 December 2023

EU Projects

ODRE: a framework to enforce ODRL policies

ODRE: a framework to enforce ODRL policies

Development

- Enforcement libraries
- Ontology extensions of ODRL
  - Time
- Derived services

Research

- Cimmino, A., Cano-Benito, J., & García-Castro, R. (2024). Open Digital Rights Enforcement Framework (ODRE): from descriptive to enforceable policies. *arXiv preprint arXiv:2409.17602*.
- Cano-Benito, J., Cimmino, A., & García-Castro, R. (2024). Towards time privacy policies in ODRL. In *NeXt-generation Data Governance workshop 2024*.
- Cano-Benito, J., Cimmino, A., & García-Castro, R. (2023, April). Injecting data into ODRL privacy policies dynamically with RDF mappings. In *Companion Proceedings of the ACM Web Conference 2023* (pp. 246-249).
- Cimmino, A., Cano-Benito, J., & García-Castro, R. (2023, April). Practical challenges of ODRL and potential courses of action. In Companion Proceedings of the ACM Web Conference 2023 (pp. 1428-1431).

ODRE: a framework to enforce ODRL policies

open source

GitHub Organization

Development

Research

OPEN ACCESS

F A I R

Findable Accessible Interoperable Reusable

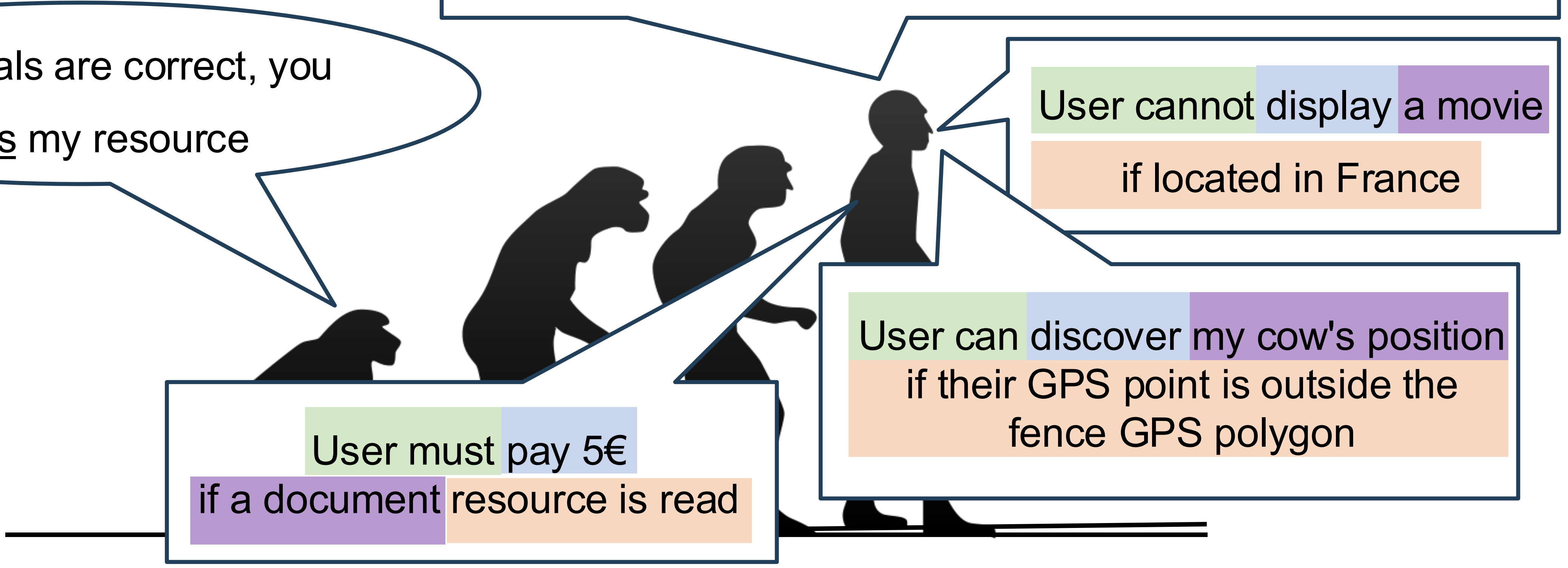
- Enforcement libraries

PIP Maven™

- Ontology extensions of ODRL
  - Time
- Derived services

- Cimmino, A., Cano-Benito, J., & García-Castro, R. (2024). Open Digital Rights Enforcement Framework (ODRE): from descriptive to enforceable policies. *arXiv preprint arXiv:2409.17602*.
- Cano-Benito, J., Cimmino, A., & García-Castro, R. (2024). Towards time privacy policies in ODRL. In *NeXt-generation Data Governance workshop 2024*.
- Cano-Benito, J., Cimmino, A., & García-Castro, R. (2023, April). Injecting data into ODRL privacy policies dynamically with RDF mappings. In *Companion Proceedings of the ACM Web Conference 2023* (pp. 246-249).
- Cimmino, A., Cano-Benito, J., & García-Castro, R. (2023, April). Practical challenges of ODRL and potential courses of action. In Companion Proceedings of the ACM Web Conference 2023 (pp. 1428-1431).

The evolution of (ODRL) Policies


If credentials are correct, you can access my resource
Under which circumstances?
Only during nighttime,
you can read my data
PERMISSION/PROHIBITITION/DUTY What? Which resource?
User cannot display a movie
if located in France
User can discover my cow's position
if their GPS point is outside the fence GPS polygon
User must pay 5€
if a document resource is read

ODRE approach to enforcement: fine-grinding the ODRL enforcement

Enforcement understanding

Ontology must define the concepts (circumstances, actions, resources, …) unambiguously for people and machines. Lexical differences must no affect the common understanding of policies, ontological terms must be clearly defined so their interpretation is closed.

Requirements from use cases

Use cases must be collected to derive requirements, problems, and practical challenges. These must be addressed by the understanding layer or the theoretical enforcement layer. *For instance, are these common or individual requirements? Can these road-blocks be addressed with technological means or only manually?*

Theoretical enforcement

Should promote an implementation-agnostic proposal; open enough (flexible) to include new practical challenges or use cases and closed enough to be implemented and used

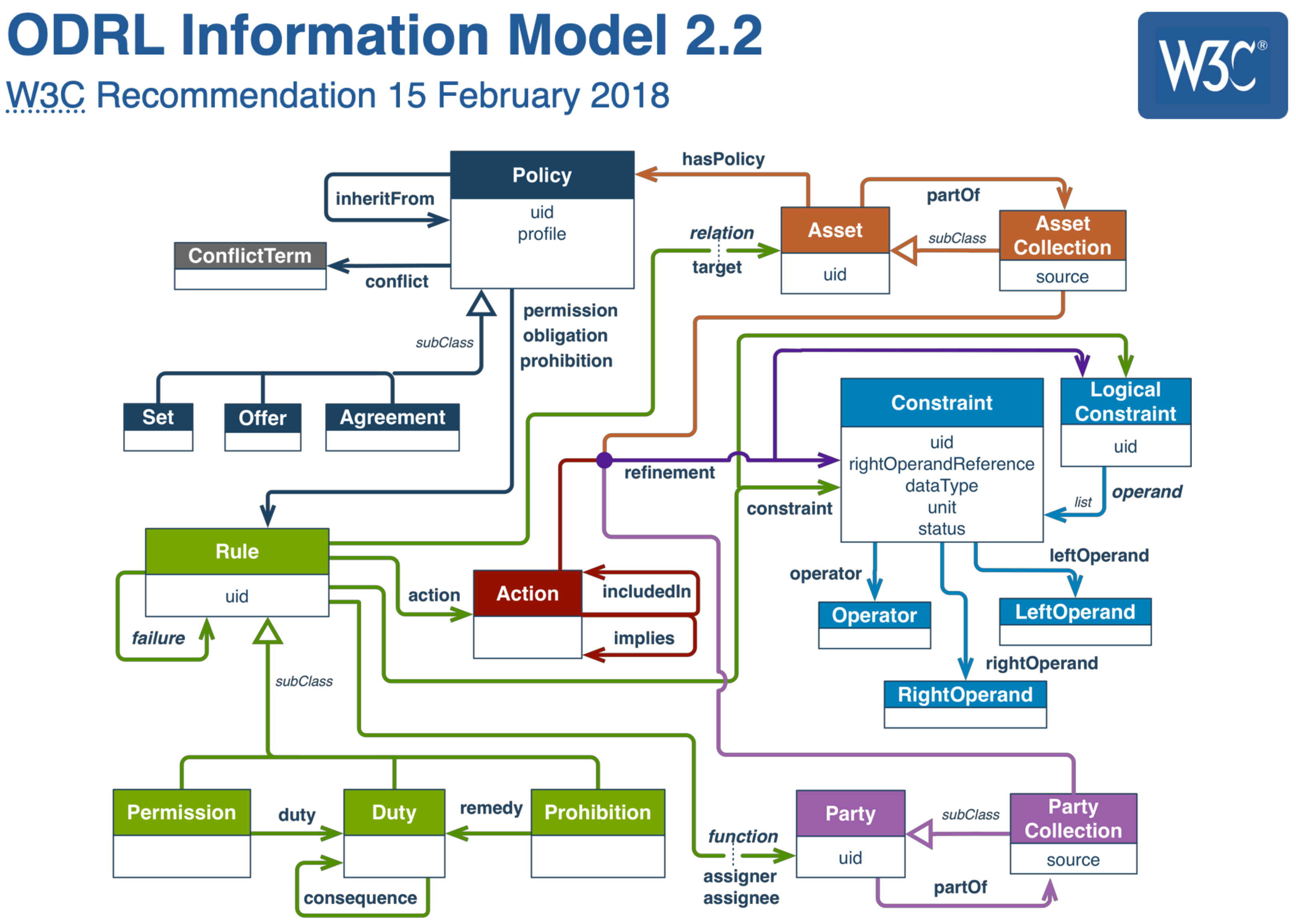
Implementations

Must meet what previous layers defined or established

ODRE approach to enforcement: fine-grinding the ODRL enforcement

Enforcement understanding	Ontology must define the concepts (circumstances, actions, resources, …) unambiguously for people and machines. Lexical differences must no affect the common understanding of policies, ontological terms must be clearly defined so their interpretation is closed.
Requirements from use cases	Use cases must be collected to derive requirements, problems, and practical challenges. These must be addressed by the understanding layer or the theoretical enforcement layer. *For instance, are these common or individual requirements? Can these road-blocks be addressed with technological means or only manually?*
Theoretical enforcement	Should promote an implementation-agnostic proposal; open enough (flexible) to include new practical challenges or use cases and closed enough to be implemented and used
Implementations	Must meet what previous layers defined or established

Enforcement Understanding: ODRL ontology



Enforcement Understanding: ODRL enforcement

P1
```
...
"constraint": {
 "leftOperand": "dateTime",
 "operator": "gt" ,
 "rightOperand": {
  "@value": "2006-05-30T09:00:00",
  "@type": "xsd:dateTime"
 }
...
```

P2
```
...
"constraint": {
 "leftOperand": "dateTime",
  "operator": "gt" ,
  "rightOperand": {
   "@value": "09:00:00",
   "@type": "xsd:time"
  }
...
```

P3
```
...
"constraint": {
 "leftOperand": {
  "@type": "schema:Person", "schema:years": "17”
 },
 "operator": "gt" ,
 "rightOperand": { "@value": "18", "@type":
"xsd:integer"}
...
```

P4
```
...
"constraint": {
 "leftOperand": { "@value": "31", "@type": "xsd:integer" },
 "operator": "gt" ,
 "rightOperand": { "@value": "18", "@type": "xsd:integer" }
...
```

- These policies use the same term, *odrl:gt,* but they add a small semantic flavour:
  - In P1 means that the current date has a value after the right’s operand value
  - In P2 means that the current date has a value after the right’s operand value, however, this later one is xsd:time and therefore they should not be compared
  - In P3 means that *a schema:Person* age is above 18 (over 18)
  - In P4 means that one whole number (W) is greater than the other.

Enforcement Understanding: ODRL enforcement

P1
```
...
"constraint": {
 "leftOperand": "dateTime",
 "operator": "gt" ,
 "rightOperand": {
  "@value": "2006-05
  "@type": "x
 }
...
```

P2
```
...
"constraint": {
 "leftOperand": "dateTime",
 "operator": "gt" ,
 "rightOperand": {
```

P3
```
...
"constraint": {
 "leftOperand": {
  "@type": "schema:Person", "schema:years": "17"
 },
 "operator": "gt" ,
 "rightOperand": { "@value": "18", "@type":
  integer"}
```

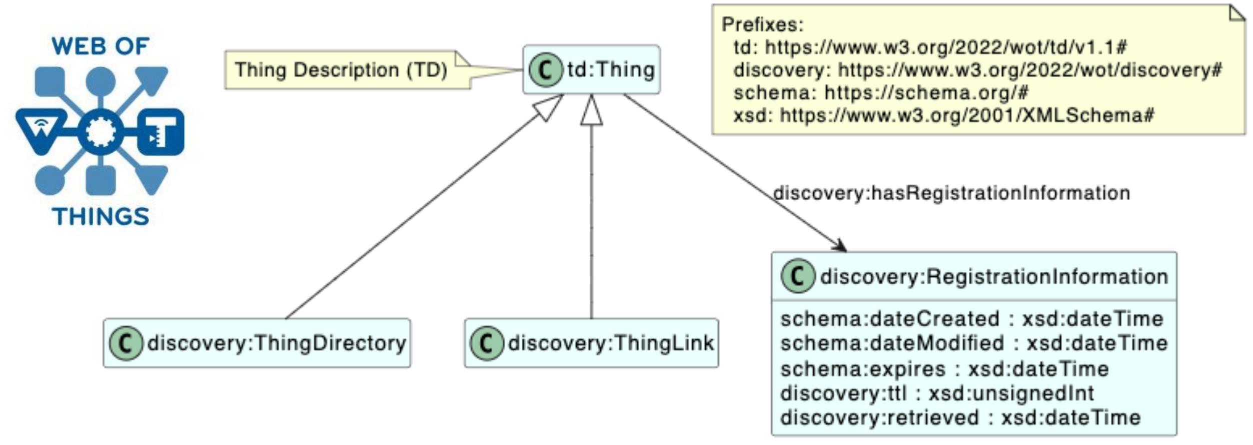
```
...
"constraint": {
 "left
 "oper
 "righ        "@value": "18", "@type": "xsd:integer" }
...
```

As humans we can infer these subtleties, however, two people may infer, and therefore understand, different meanings. The context helps, for instance, P3 and P4 may aim at representing if someone is above 18

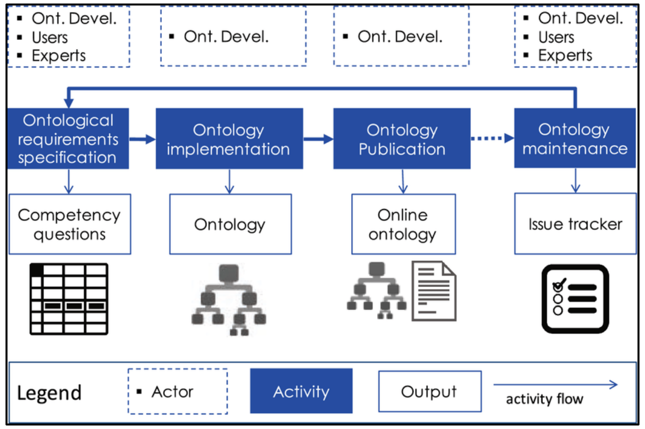
- Th use the same term, *odrl:gt,* but they add a small avour:
  - In P1 means that th
  - In P2 means that th ever, this later one is xsd:time and therefore they
  - In P3 means that a
  - In P4 means that one whole number (W) is greater than the other.

As machine these subtleties ~~may~~ led to different ways of processing data, i.e., different ways of implementing this operator

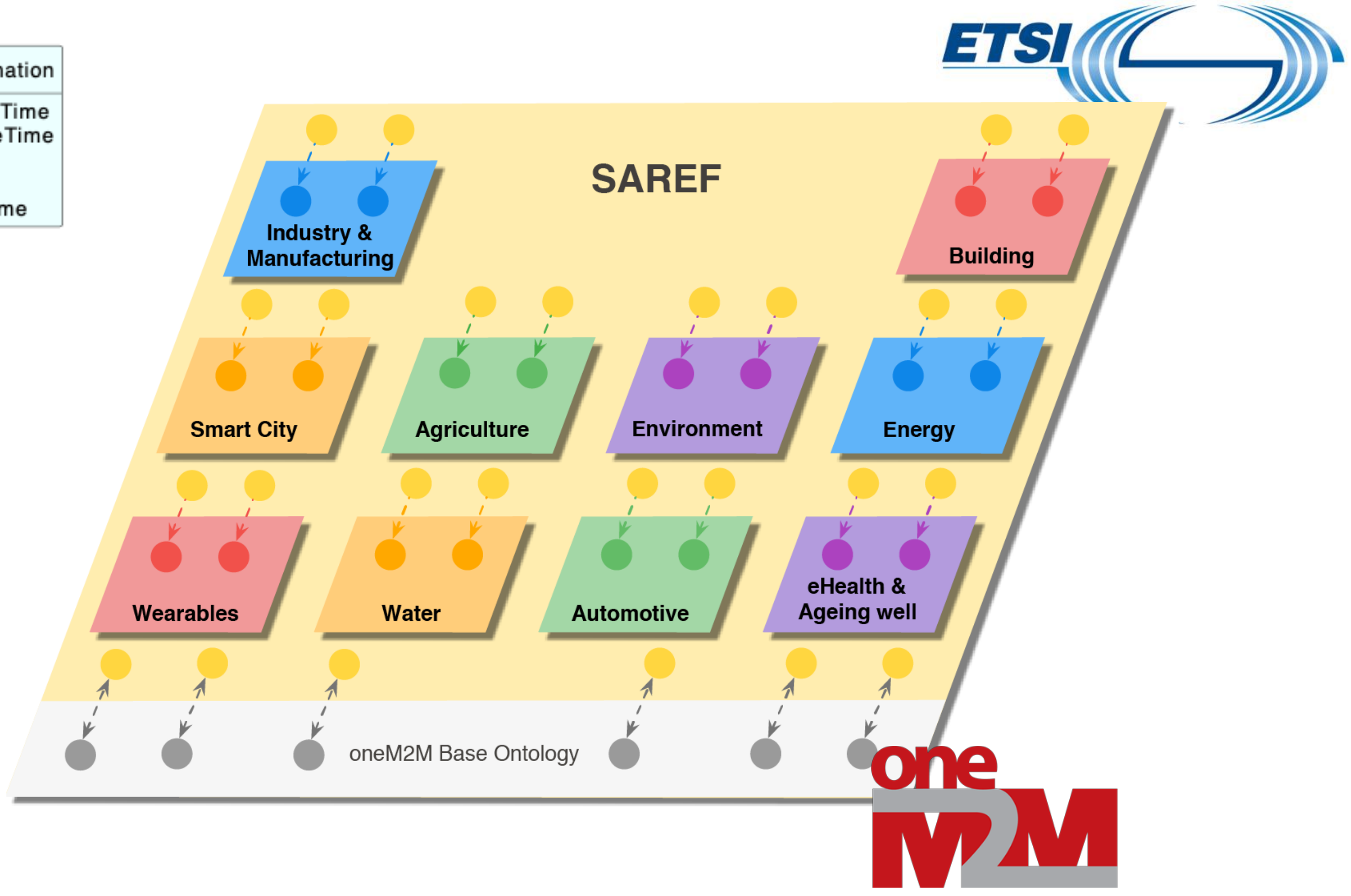
How other standards are addressing this issue


WEB OF THINGS
Thing Description (TD)
td:Thing
discovery:hasRegistrationInformation
discovery:ThingDirectory
discovery:ThingLink
discovery:RegistrationInformation
schema:dateCreated : xsd:dateTime
schema:dateModified : xsd:dateTime
schema:expires : xsd:dateTime
discovery:ttl : xsd:unsignedInt
discovery:retrieved : xsd:dateTime

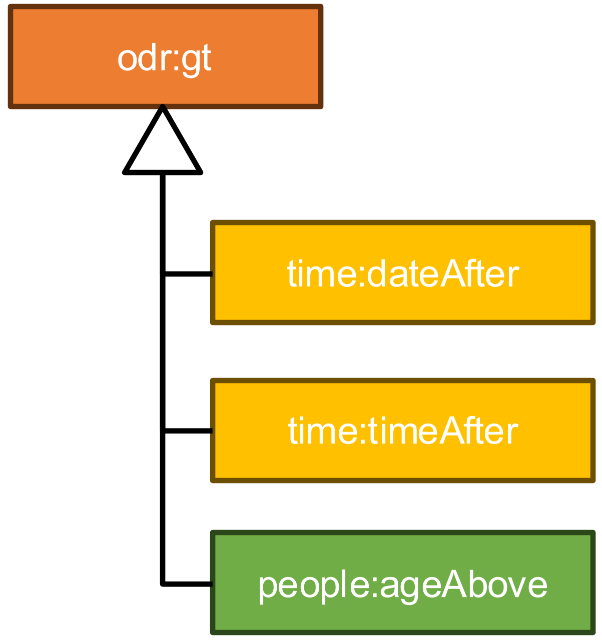
Prefixes:
td: https://www.w3.org/2022/wot/td/v1.1#
discovery: https://www.w3.org/2022/wot/discovery#
schema: https://schema.org/#
xsd: https://www.w3.org/2001/XMLSchema#



LoT methodology



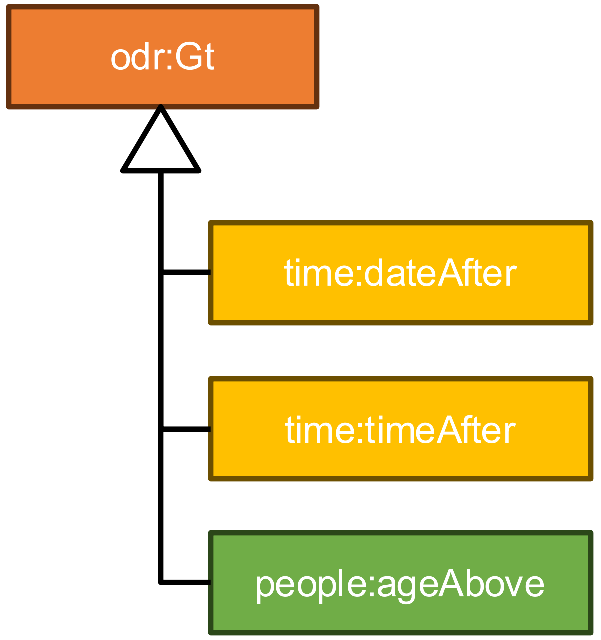
Enforcement Understanding: domain-specific modules

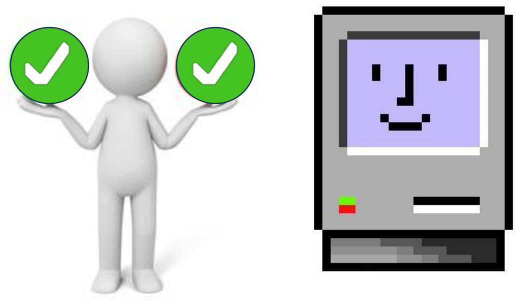


- Allows narrowing down the definition of terms and concepts, reducing potential interpretations and increasing their unambiguous understanding:
  - For example, “time:dateAfter” compares two xsd:dateTime operands, cannot compare operands of different datatypes.
- Provide domain-specific artefacts related to the ontology:
  - Documentation
  - Ontology portal
  - SHACL shapes
  - JSON-LD 1.1 contexts
  - …

THESE MODULES SHOULD COME FROM THE STANDARD

Enforcement Understanding: domain-specific modules





```
...
"constraint": {
 "leftOperand": { "@value": "31", "@type": "xsd:integer" },
 "operator": "gt" ,
 "rightOperand": { "@value": "18", "@type": "xsd:integer" }
...
```

```
...
"constraint": {
 "leftOperand": "dateTime",
 "operator": "time:dateAfter" ,
 "rightOperand": {
  "@value": "2002-05-30T09:00:00",
  "@type": "xsd:dateTime"
 }
...
```

```
...
"constraint": {
 "leftOperand": "dateTime",
  "operator": "time:timeAfter" ,
  "rightOperand": {
   "@value": "09:00:00",
   "@type": "xsd:time"
  }
...
```

```
...
"constraint": {
 "leftOperand": {
    "@type": "schema:Person", "schema:years" : "17”
  },
 "operator": "people:ageAbove" ,
 "rightOperand": {
    "@value": "18", "@type": "xsd:integer"
 }
...
```

Take away

- It is to be **expected to have multiple implementations** for the **enforcement**. Even for the same action, operand or operator.
- It is important that the **different implementations do not differ in the abstract behaviour** just "*in the code".*
    - Somehow, the **ontological term should act as an interface** that the code implements
- **Defining domain-specific ontologies lower** the chances of **interpreting differently the same ontological concept** by narrowing down its meaning
    - This approach reduces the divergence of potential enforcement implementations
- Having a core ontology that is more abstract, and domain specific extensions is a common practice in ontology engineering

ODRE approach to enforcement: fine-grinding the ODRL enforcement

Enforcement understanding

Ontology must define the concepts (circumstances, actions, resources, …) unambiguously for people and machines. Lexical differences must no affect the common understanding of policies, ontological terms must be clearly defined so their interpretation is closed.

Requirements from use cases

Use cases must be collected to derive requirements, problems, and practical challenges. These must be addressed by the understanding layer or the theoretical enforcement layer. *For instance, are these common or individual requirements? Can these road-blocks be addressed with technological means or only manually?*

Theoretical enforcement

Should promote an implementation-agnostic proposal; open enough (flexible) to include new practical challenges or use cases and closed enough to be implemented and used

Implementations

Must meet what previous layers defined or established

Requirements, Use cases, applications,….

- Architecture for Unified Regional and Open digital ecosystems for Smart Communities and wider Rural Areas Large scale application → AURORAL (101016854) / doi 10.3030/101016854
- H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT)
- Budget 16.265.962,26€
- Partners: 30 partners
- 8 pilots
- More tan 20 cross-domain use cases

https://www.auroral.eu

Requirements extracted

- *R1: Dynamic data must be handled in policies during enforcement*
  - Policies rely on operands holding data:
    - Written in the policy → may lead to privacy leak HARD RESTICTION
    - Known by the enforcement system SOFT RESTICTION
    - Provided by someone (whoever is enforcing the policy) SOFT RESTICTION
    - Provided by a third-party service SOFT RESTICTION
- *R2: Enforcement should specify if an action is part of the enforcement or must be performed by someone*
  - Actions are also something that must ”happen” during enforcement
  - In the case action is a physical action, who has such responsibility

Requirements extracted

- *R3: Enforcement must support and couple the ontological terms with code implementations, for this, a well and closed definition of the terms is needed*
  - Ontology extensions serve to this purpose
- *R4: New terms in the ontology may appear, or extensions of the ontology, and the enforcement must be able to adapt to them by including new behaviour coupled with such terms.*
- *R5: Enforcement must support N-ary operands and operators*
  - Order in the arguments may be challenging
- *R6: Enforcement must be synchronous and/or asynchronous*
  - Maybe this information should be specified in the policy
- *R7: Compatibility among operands in an operator must be considered*
  - We cannot compare xsd:date with xsd:time, xsd:dateTime with xsd:date, ….

Take away

- ODRE has collected different requirements from the use cases of the project AURORAL
    - The collection of **use cases if coming from the standard** will **improve by far what we need (the world)** to take into consideration **in the enforcement** process
- **ODRE implementation does not cover all these requirements** since some (like R5 or R6) **move too away from the actual standard**
- We have **stated only** what we consider the **most important requirements, not all those collected**

ODRE approach to enforcement: fine-grinding the ODRL enforcement

Enforcement understanding

Ontology must define the concepts (circumstances, actions, resources, …) unambiguously for people and machines. Lexical differences must no affect the common understanding of policies, ontological terms must be clearly defined so their interpretation is closed.

Requirements from use cases

Use cases must be collected to derive requirements, problems, and practical challenges. These must be addressed by the understanding layer or the theoretical enforcement layer. *For instance, are these common or individual requirements? Can these road-blocks be addressed with technological means or only manually?*

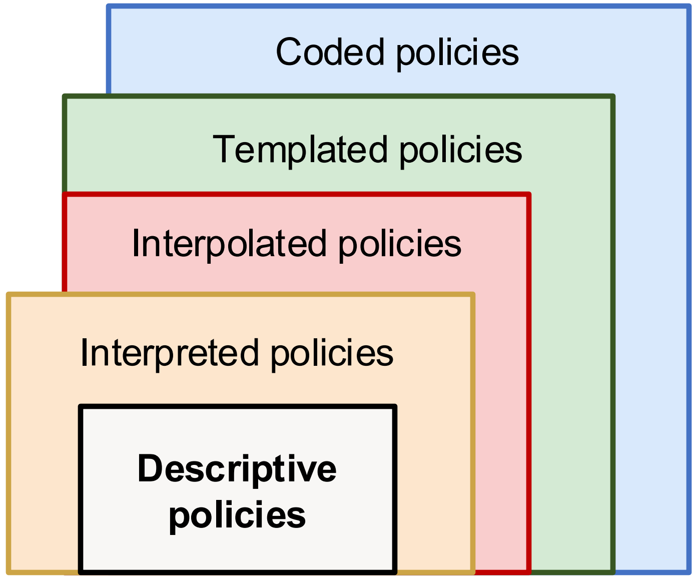
Theoretical enforcement

Should promote an implementation-agnostic proposal; open enough (flexible) to include new practical challenges or use cases and closed enough to be implemented and used

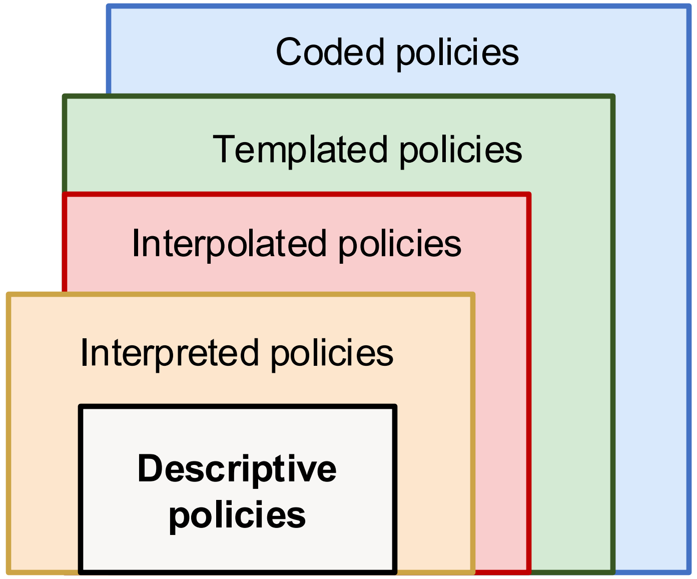
Implementations

Must meet what previous layers defined or established

Theoretical enforcement: higher abstract policies

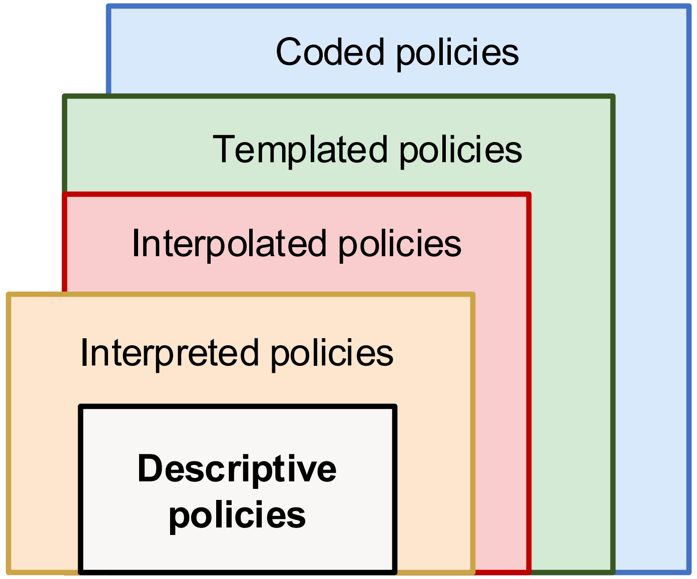


Theoretical enforcement: higher abstract policies



These are **policies** that are expressed **with terms from the ODRL ontology or extensions of it**. As a result, **they cannot be enforced**.

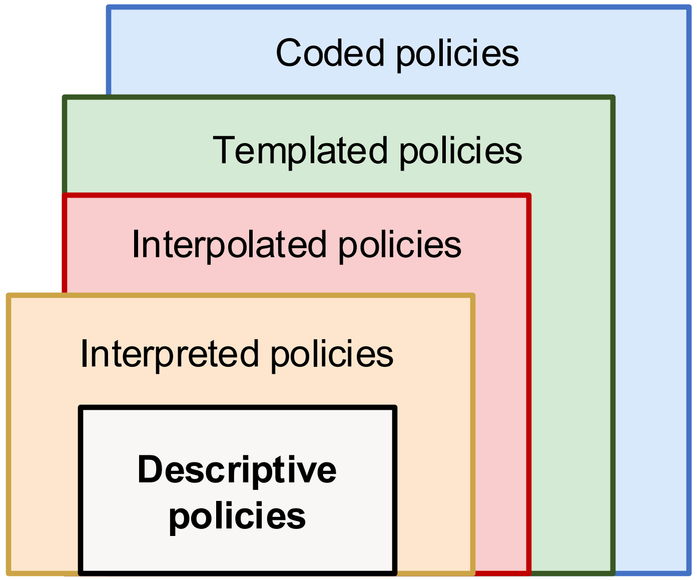
Theoretical enforcement: higher abstract policies



These are **policies** that are expressed **with terms coupled with some code**. As a result, **they can be enforced**. Nevertheless, they **only support hard-constrains and soft constrains using ODRL operands.**

These are **policies** that are expressed **with terms from the ODRL ontology or extensions of it**. As a result, **they cannot be enforced**.

Theoretical enforcement: higher abstract policies

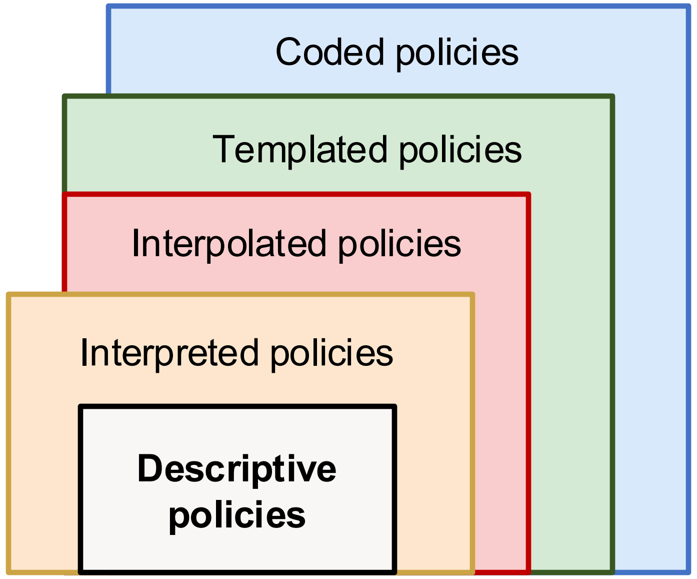


These are **policies** that are expressed **mixing an interpolation language with RDF**. Interpolation language **allows to define data variables** in the policy **that are replaced** with a certain value **during enforcement. Meant to handle data provided by a requester**

These are **policies** that are expressed **with terms coupled with some code**. As a result, **they can be enforced**. Nevertheless, they **only support hard-constrains and soft constrains using ODRL operands.**

These are **policies** that are expressed **with terms from the ODRL ontology or extensions of it**. As a result, **they cannot be enforced**.

Theoretical enforcement: higher abstract policies



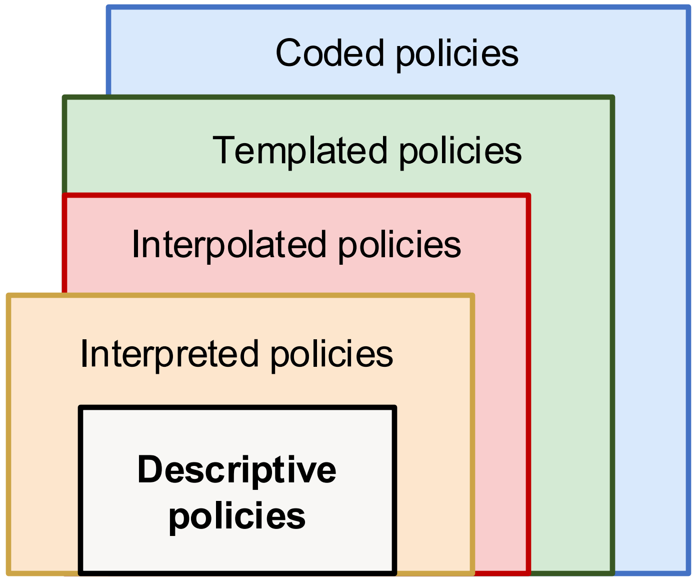
These are **policies** that are expressed **mixing a templated language with RDF**. This language extends interpolated one with control flow, functions definitions, etc. For instance, an HTML template language.

These are **policies** that are expressed **mixing an interpolation language with RDF**. Interpolation language **allows to define data variables** in the policy **that are replaced** with a certain value **during enforcement. Meant to handle data provided by a requester**

These are **policies** that are expressed **with terms coupled with some code**. As a result, **they can be enforced**. Nevertheless, they **only support hard-constrains and soft constrains using ODRL operands.**

These are **policies** that are expressed **with terms from the ODRL ontology or extensions of it**. As a result, **they cannot be enforced**.

Theoretical enforcement: higher abstract policies



These are **policies** that are expressed **mixing a templated language with RDF**. However, the **template language references functions** that are implemented **with a code from a programming language**, e.g., java, and that invoked during enforcement.

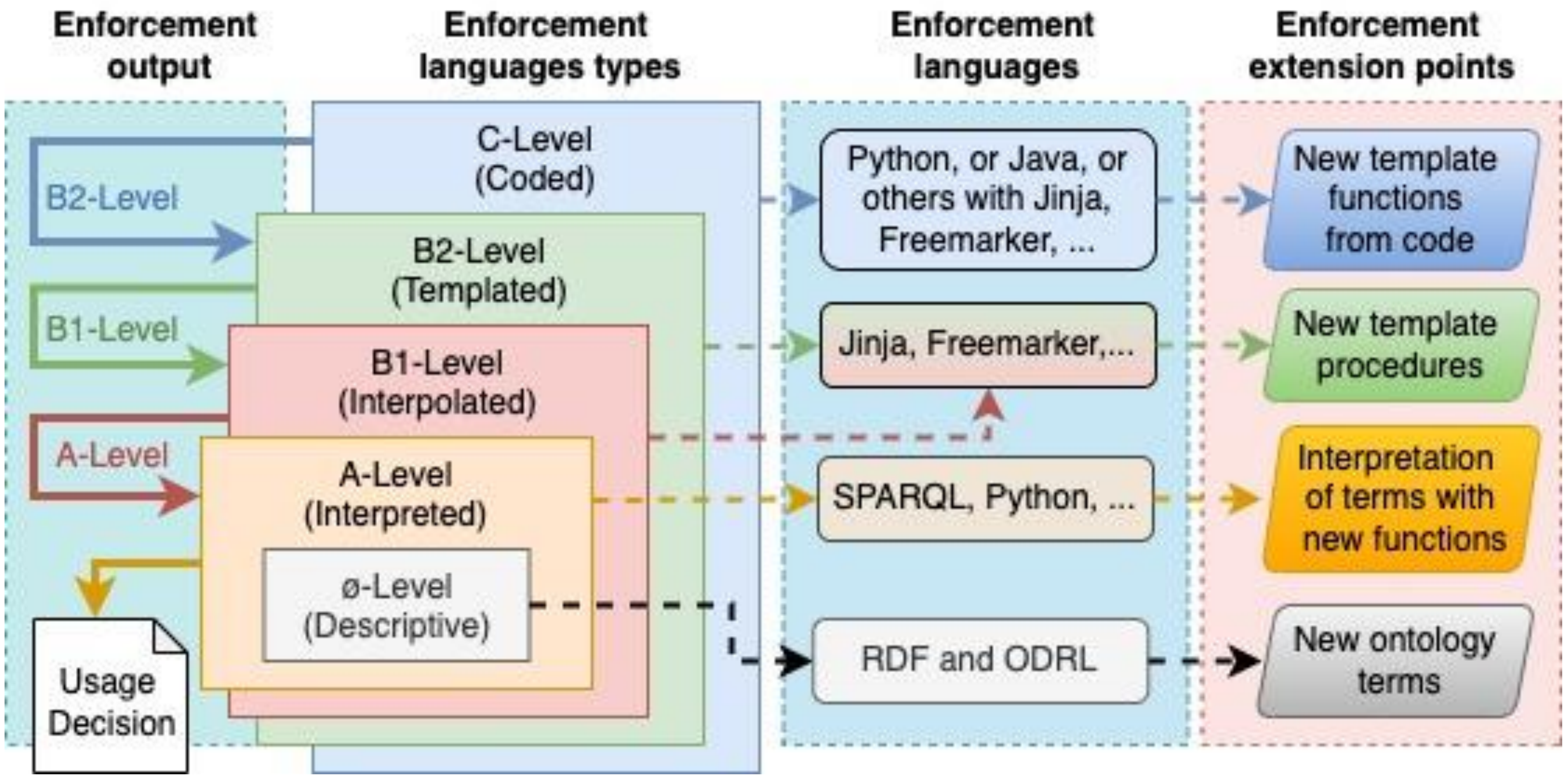
These are **policies** that are expressed **mixing a templated language with RDF**. This language extends interpolated one with control flow, functions definitions, etc. For instance, an HTML template language.

These are **policies** that are expressed **mixing an interpolation language with RDF**. Interpolation language **allows to define data variables** in the policy **that are replaced** with a certain value **during enforcement. Meant to handle data provided by a requester**

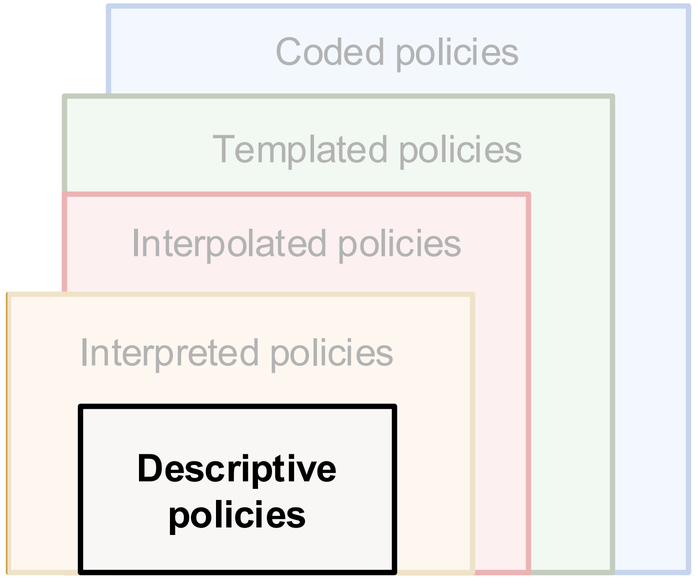
These are **policies** that are expressed **with terms coupled with some code**. As a result, **they can be enforced**. Nevertheless, they **only support hard-constrains and soft constrains using ODRL operands.**

These are **policies** that are expressed **with terms from the ODRL ontology or extensions of it**. As a result, **they cannot be enforced**.

Theoretical enforcement: higher abstract policies

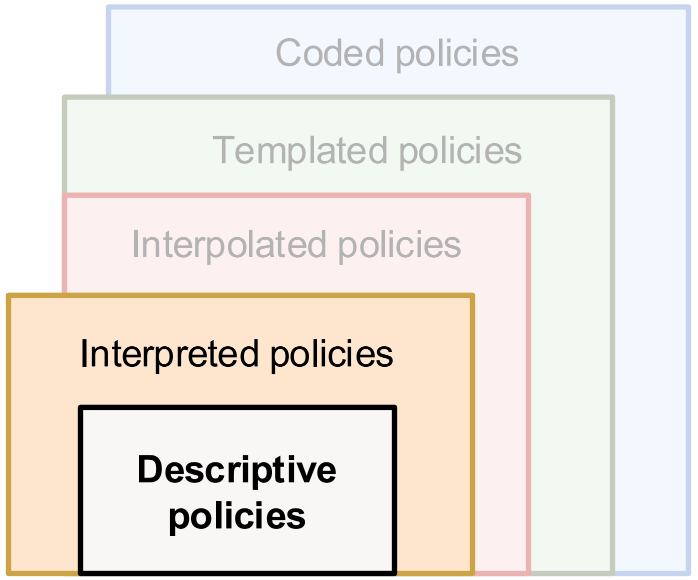


Descriptive Policy



```
{
  "@context": "...",
  "@type": "Policy",
  "uid": "https://upm.es/policy/1",
  "permission": [
    {
      "target": "https://jsonplaceholder.typicode.com/users/1",
      "action": "read",
      "constraint": [
        {
          "leftOperand": "dateTime",
          "operator": "gt",
          "rightOperand": {
            "@value": "2018-01-01T00:40:30",
            "@type": "xsd:dateTime"
          }
        },
        {
          "leftOperand": "media",
          "operator": "eq",
          "rightOperand": {
            "@value": "online",
            "@type": "xsd:string"
          }
        }]}]}
```

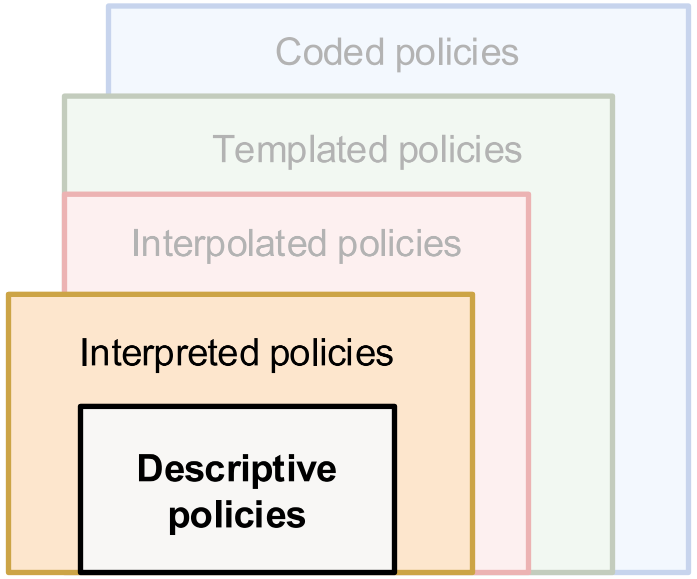
Interpreted Policy



```
{
  "@context": "...",
  "@type": "Policy",
  "uid": "https://upm.es/policy/1",
  "permission": [
    {
      "target": "https://jsonplaceholder.typicode.com/users/1",
      "action": "read",
      "constraint": [
        {
          "leftOperand": "dateTime",
          "operator": "gt",
          "rightOperand": {
            "@value": "2018-01-01T00:40:30",
            "@type": "xsd:dateTime"
          }
        },
        {
          "leftOperand": "media",
          "operator": "eq",
          "rightOperand": {
            "@value": "online",
            "@type": "xsd:string"
          }
        }]}]}
```

odrl_gt(odrl_datetime(), cast_dateTime("2018-01-01T00:40:30")) and odrl_eq(odrl_media(), cast_string("online"))

```
  "uid": "https://upm.es/policy/1",
  "permission": [
    {
      "target": "https://jsonplaceholder.typicode.com/users/1",
      "action": "read",
      "constraint": [
        {
          "leftOperand": "dateTime",
          "operator": "gt",
          "rightOperand": {
            "@value": "2018-01-01T00:40:30",
            "@type": "xsd:dateTime"
          }
        },
        {
          "leftOperand": "media",
          "operator": "eq",
          "rightOperand": {
            "@value": "online",
            "@type": "xsd:string"
          }
        }]}]}
```



Interpreted Policy

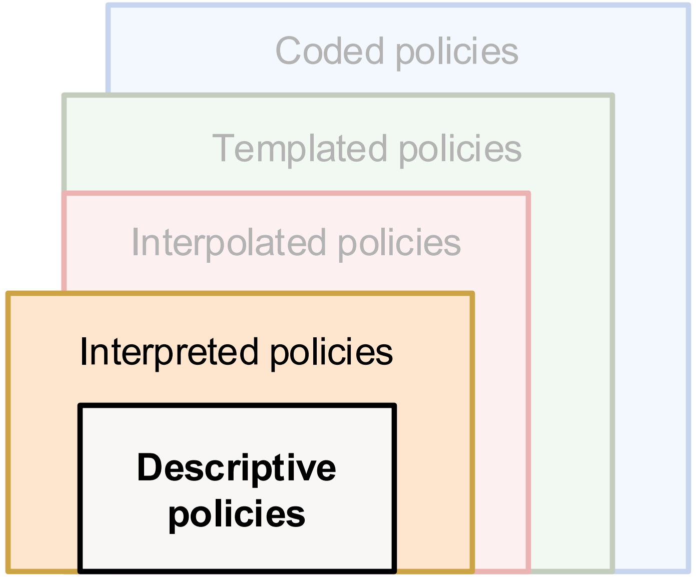
```
odrl_gt(odrl_datetime(), cast_dateTime("2018-01-01T00:40:30")) and odrl_eq(odrl_media(), cast_string("online"))
```

```
SELECT ?enforcementResult {
        BIND (
                fnc:now() > xsd:dateTime("2018-01-01T00:40:30")
                AND
                fnc:media() == xsd:string("online")
                AS ?enforcementResult
        ) .}
```

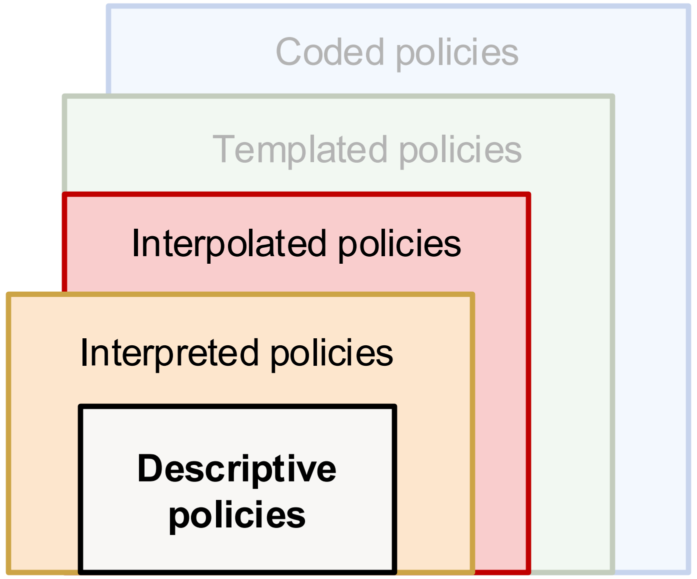
```
                    ://upm.es/policy/1",
                     [

                      "https://jsonplaceholder.typicode.com/users/1",
                      "read",
                    nt": [

                    Operand": "dateTime",
        "operator": "gt",
        "rightOperand": {
          "@value": "2018-01-01T00:40:30",
          "@type": "xsd:dateTime"
        }
      },
      {
        "leftOperand": "media",
        "operator": "eq",
        "rightOperand": {
          "@value": "online",
          "@type": "xsd:string"
        }
      }]}]}
```

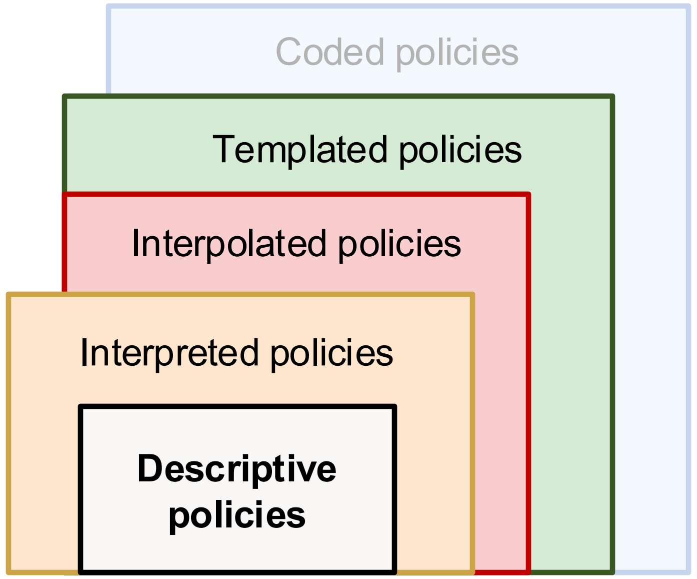


Interpreted Policy



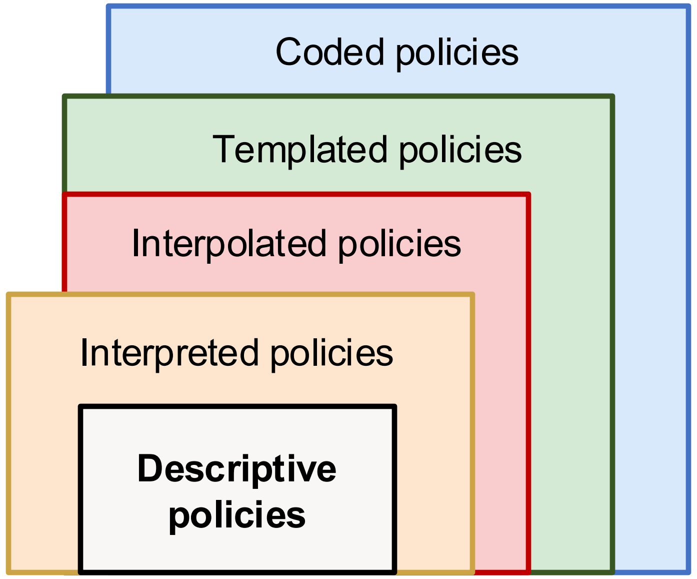
```
{
  "@context": "...",
  "@type": "Policy",
  "uid": "https://upm.es/policy/1",
  "permission": [
    {
      "target": "https://jsonplaceholder.typicode.com/users/1",
      "action": "read",
      "constraint": [
        {
          "leftOperand": "[=known_datetime]",
          "operator": "gt",
          "rightOperand": {
            "@value": "2018-01-01T00:40:30",
            "@type": "xsd:dateTime"
          }
        },
        {
          "leftOperand": "media",
          "operator": "eq",
          "rightOperand": {
            "@value": "online",
            "@type": "xsd:string"
          }
        }]}]}
```

Templated Policy



```
{
  "@context": "...",
  "@type": "Policy",
  "uid": "https://upm.es/policy/1",
  "permission": [
    {
      "target": "https://jsonplaceholder.typicode.com/users/1",
      "action": "read",
      "constraint": [
        {
[#assign timeVar=.now?time?iso("Europe/Rome")?replace('\\+.*','', 'r')]
          "leftOperand": "[=timeVar]",
          "operator": "gt",
          "rightOperand": {
            "@value": "2018-01-01T00:40:30",
            "@type": "xsd:dateTime"
          }
        },
        {
          "leftOperand": "media",
          "operator": "eq",
          "rightOperand": {
            "@value": "online",
            "@type": "xsd:string"
          }}]}]}
```

Coded Policy

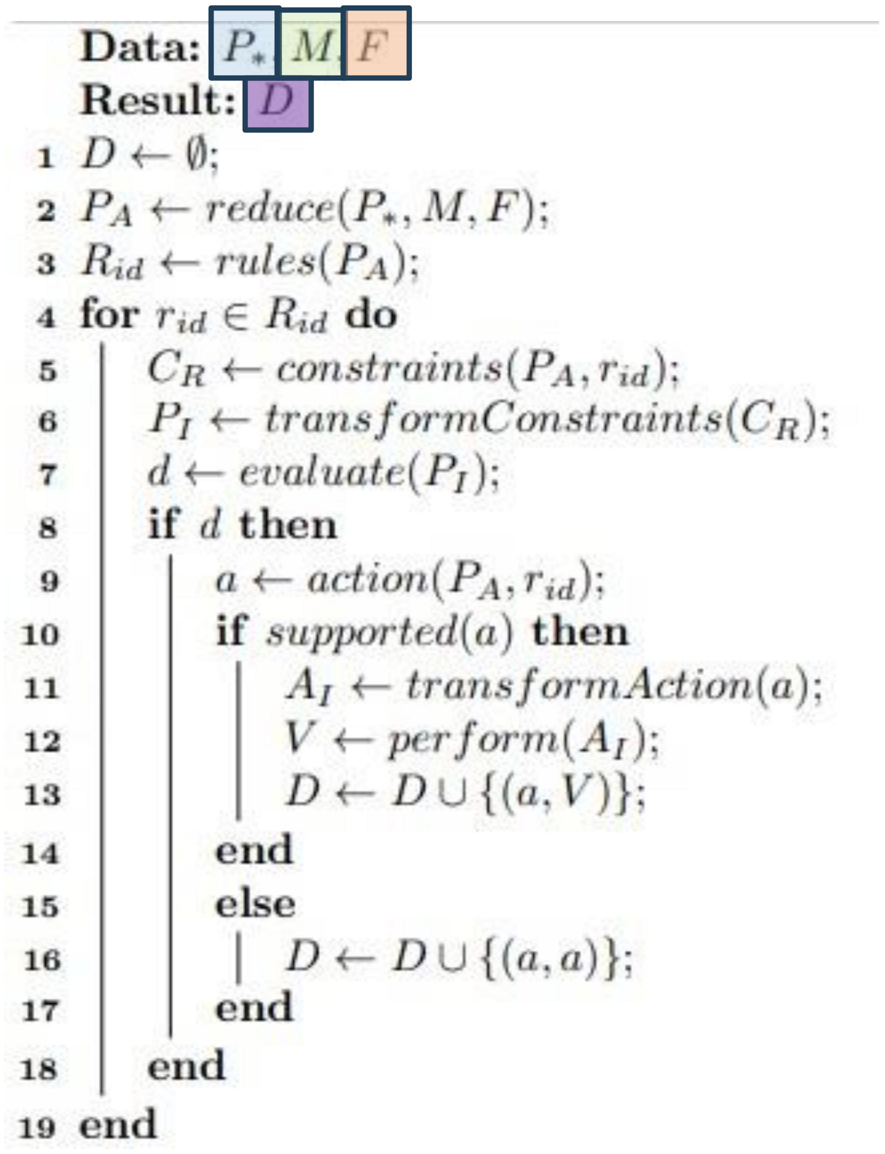


```
[#-- Data access --]
[#assign weather=request("GET", https://api.open-meteo.com/v1/forecast?
latitude=40.4050099&longitude=-3.839519&hourly=temperature_2m)]
[#-- Data handling --]
[#assign tmp="ERROR"]
[#list weather.hourly.time as elem]
[#if elem?contains("T"+currentTime) || elem?contains("T0"+currentTime)]
[#assign timeVar =weather.hourly.time[elem?index]]
[#break]
[/#if]
[/#list]
{
  "@context": "...", "@type": "Policy", "uid": "https://upm.es/policy/1",
  "permission": [{
      "target": "https://jsonplaceholder.typicode.com/users/1",
      "action": "read",
      "constraint": [
        {
          "leftOperand": "[=timeVar]",
          "operator": "gt", "rightOperand": { "@value": "2018-01-01T00:40:30",
"@type": "xsd:dateTime” }
        }, {
          "leftOperand": "media", "operator": "eq",
          "rightOperand": {
            "@value": "online", "@type": "xsd:string" }}]}]}
```

Theoretical enforcement: algorithm

```
Data: $P_*, M, F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

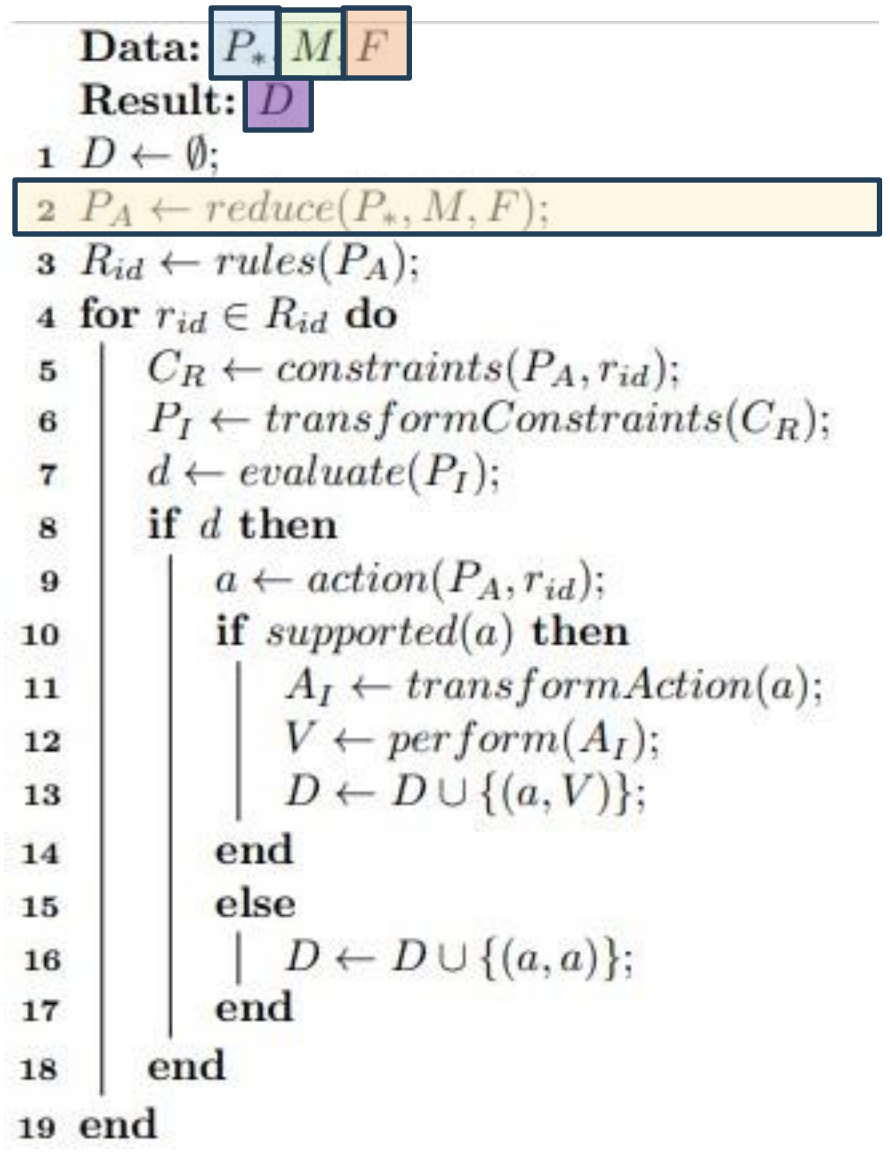
Theoretical enforcement: algorithm



```
Data: $P_*$, $M$, $F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

- P_* a ODRL policy
- M a key-value set with interpolation variables
- F a key-value set with coded functions
- Decision result after evaluation

Theoretical enforcement: algorithm

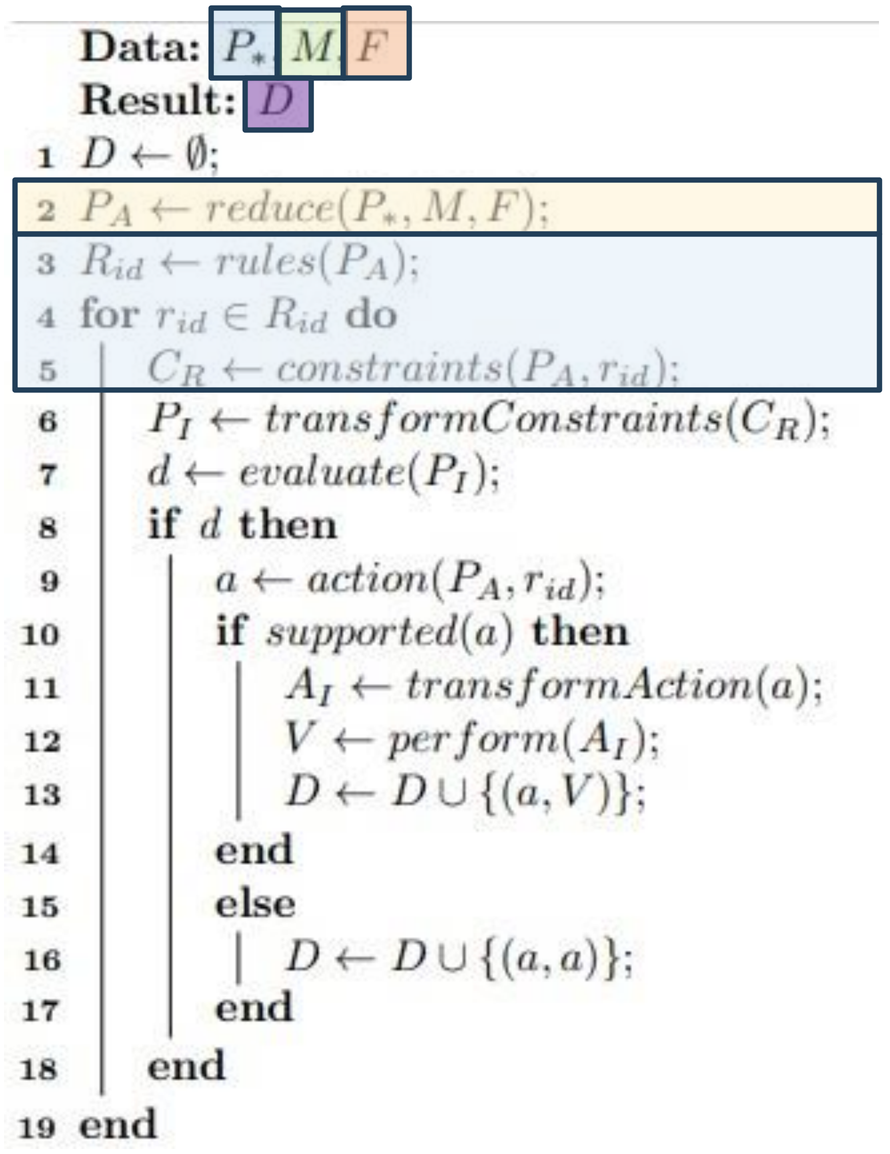


```
Data: $P_*$, $M$, $F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

- P_* a ODRL policy
- M a key-value set with interpolation variables
- F a key-value set with coded functions
- Decision result after evaluation

1. Reduce policies to interpreted language
    - inject interpolation data variables and coded function, then, solve policy
    - The result must be an interpreted policy

Theoretical enforcement: algorithm

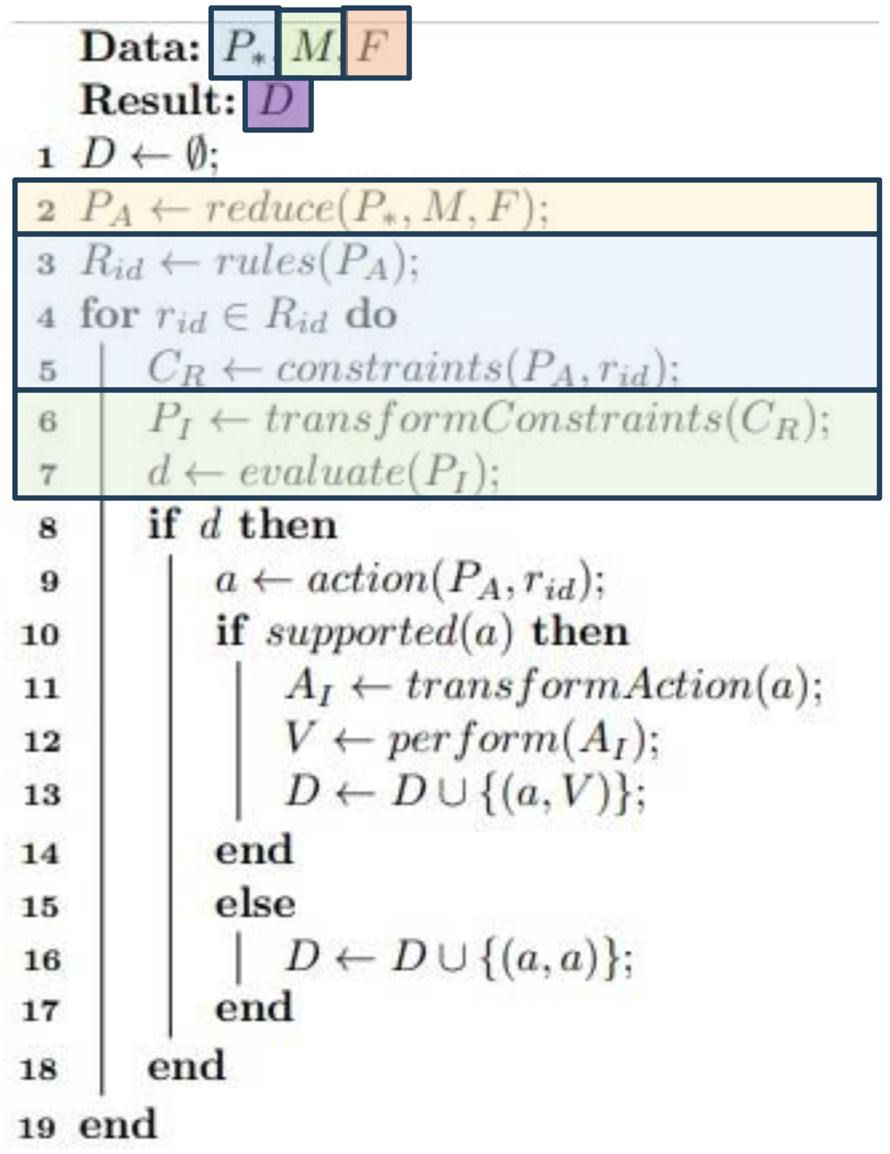


```
Data: $P_*$, $M$, $F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

- P_* a ODRL policy
- M a key-value set with interpolation variables
- F a key-value set with coded functions
- Decision result after evaluation

2. For the policy, find the rules and its constraints
    - Filter the rule using, for instance, SPARQL

Theoretical enforcement: algorithm

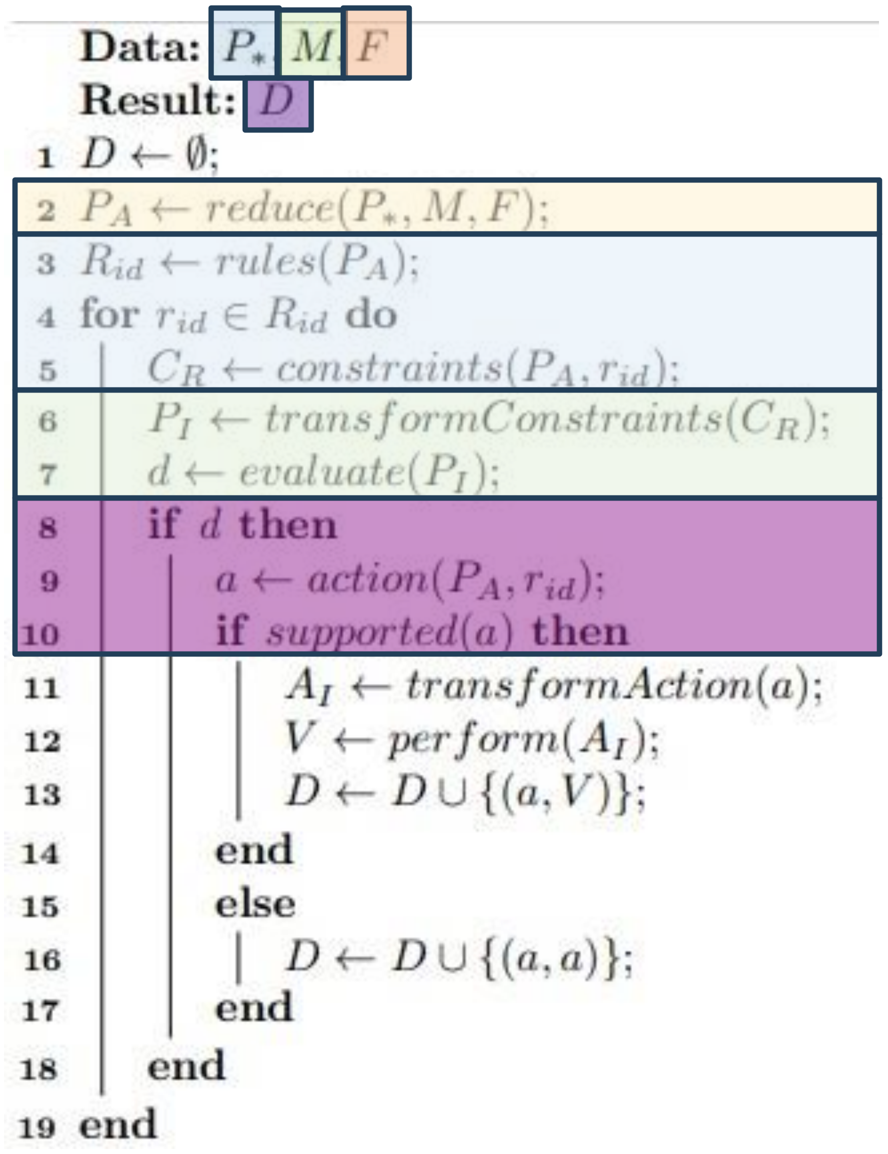


```
Data: $P_*$, $M$, $F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

- P_* a ODRL policy
- M a key-value set with interpolation variables
- F a key-value set with coded functions
- Decision result after evaluation

3. Transform the policy into an interpreted language and enforce the result

Theoretical enforcement: algorithm

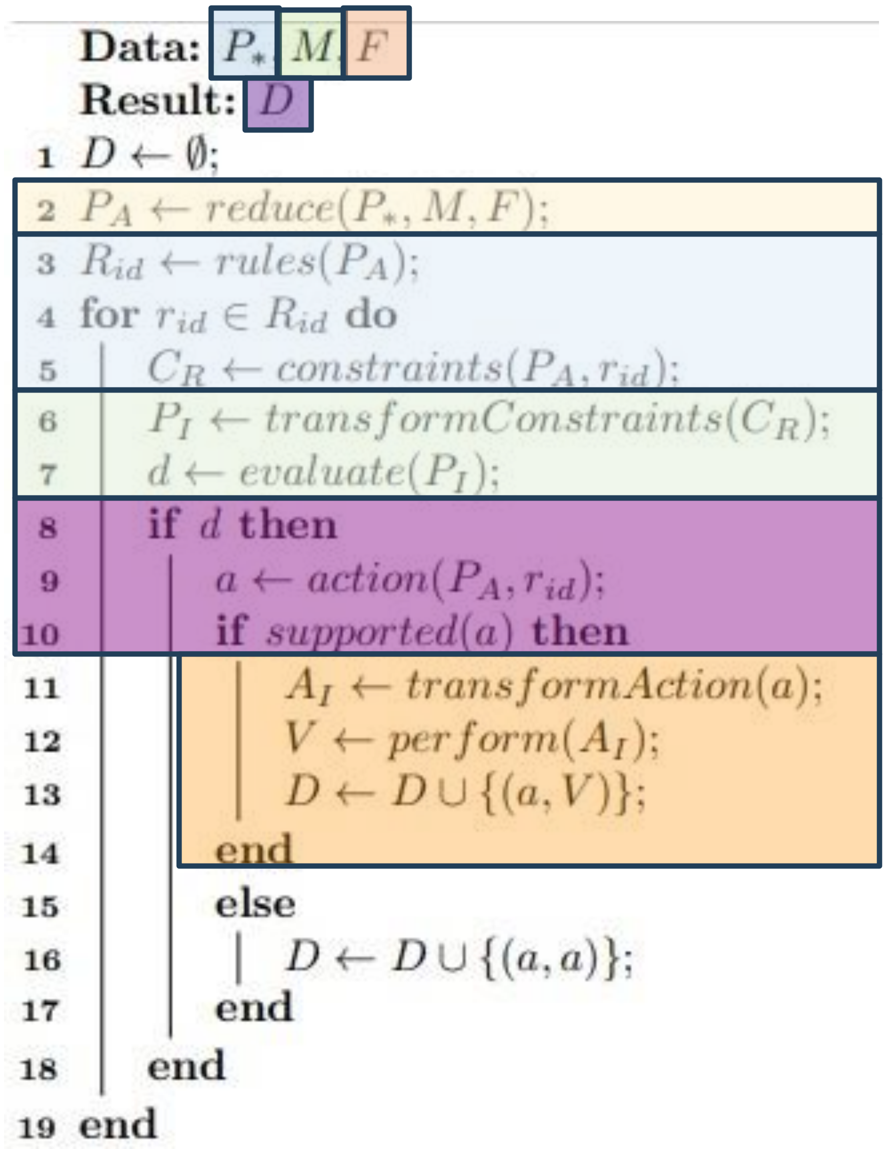


```
Data: $P_*$, $M$, $F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

- P_* a ODRL policy
- M a key-value set with interpolation variables
- F a key-value set with coded functions
- Decision result after evaluation

4. If the decision is positive, *d*, retrieve the action related to the constraints. Check if that action is enforceable, i.e., is coupled the action with some interpretable code

Theoretical enforcement: algorithm



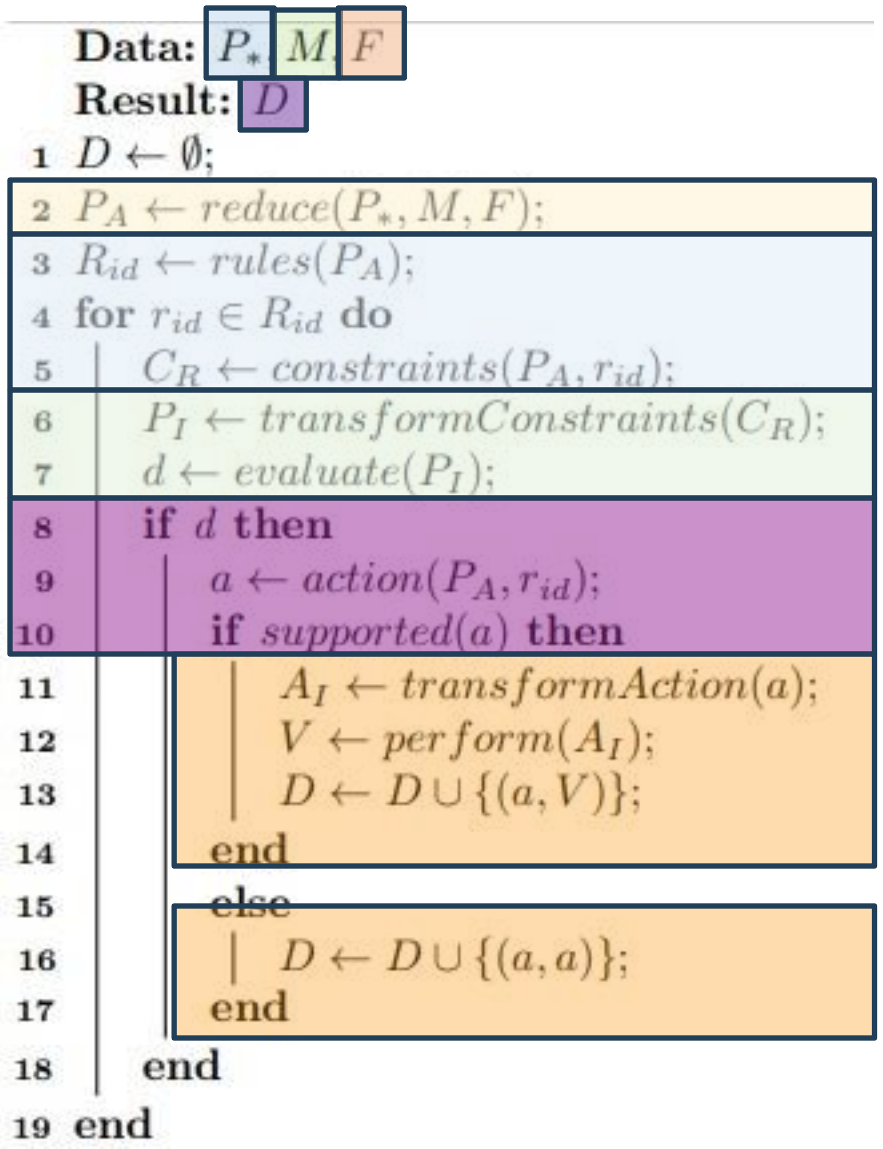
```
Data: $P_*$, $M$, $F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

- P_* a ODRL policy
- M a key-value set with interpolation variables
- F a key-value set with coded functions
- Decision result after evaluation

4. If the decision is positive, *d*, retrieve the action related to the constraints. Check if that action is enforceable, i.e., is coupled the action with some interpretable code

   4.1 Transform the action into interpretable code, run it, and store the result as part of the decision

Theoretical enforcement: algorithm



```
Data: $P_*$, $M$, $F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

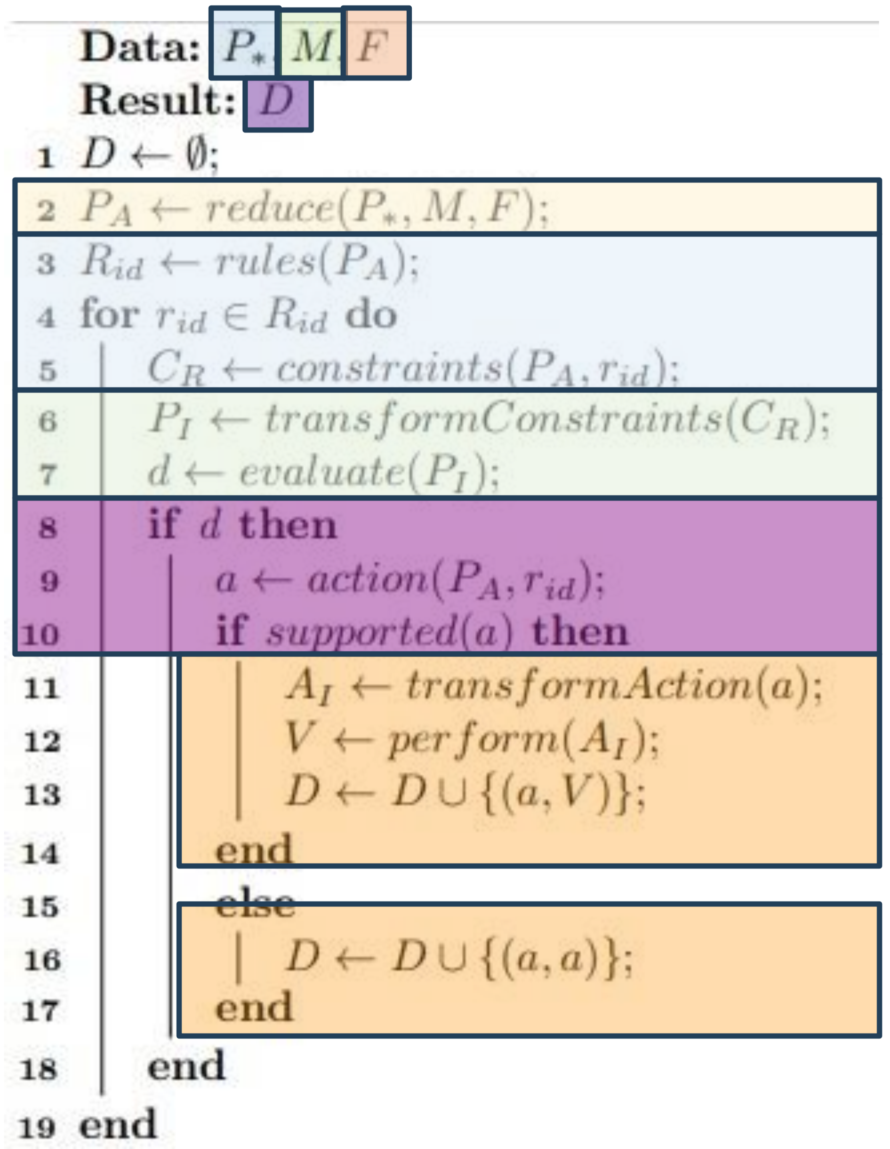
- P_* a ODRL policy
- M a key-value set with interpolation variables
- F a key-value set with coded functions
- Decision result after evaluation

4. If the decision is positive, *d*, retrieve the action related to the constraints. Check if that action is enforceable, i.e., is coupled the action with some interpretable code

    4.1 Transform the action into interpretable code, run it, and store the result as part of the decision

    4.2 Store the action URI as part of the decision

Theoretical enforcement: algorithm

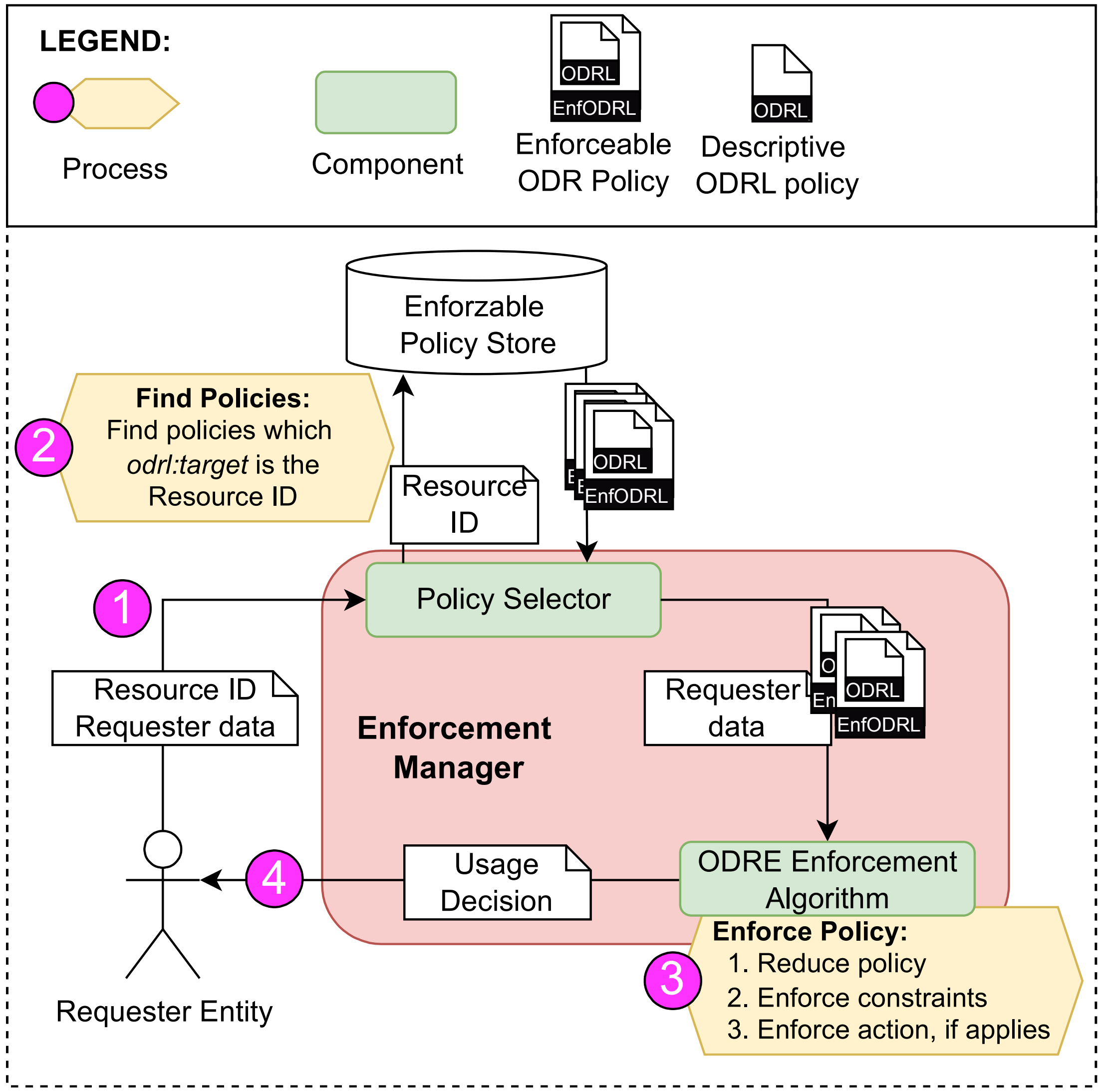


```
Data: $P_*$, $M$, $F$
Result: $D$
$D \leftarrow \emptyset$;
$P_A \leftarrow reduce(P_*, M, F)$;
$R_{id} \leftarrow rules(P_A)$;
for $r_{id} \in R_{id}$ do
    $C_R \leftarrow constraints(P_A, r_{id})$;
    $P_I \leftarrow transformConstraints(C_R)$;
    $d \leftarrow evaluate(P_I)$;
    if $d$ then
        $a \leftarrow action(P_A, r_{id})$;
        if $supported(a)$ then
            $A_I \leftarrow transformAction(a)$;
            $V \leftarrow perform(A_I)$;
            $D \leftarrow D \cup \{(a, V)\}$;
        end
        else
            $D \leftarrow D \cup \{(a, a)\}$;
        end
    end
end
```

- P_* a ODRL policy
- M a key-value set with interpolation variables
- F a key-value set with coded functions
- Decision result after evaluation

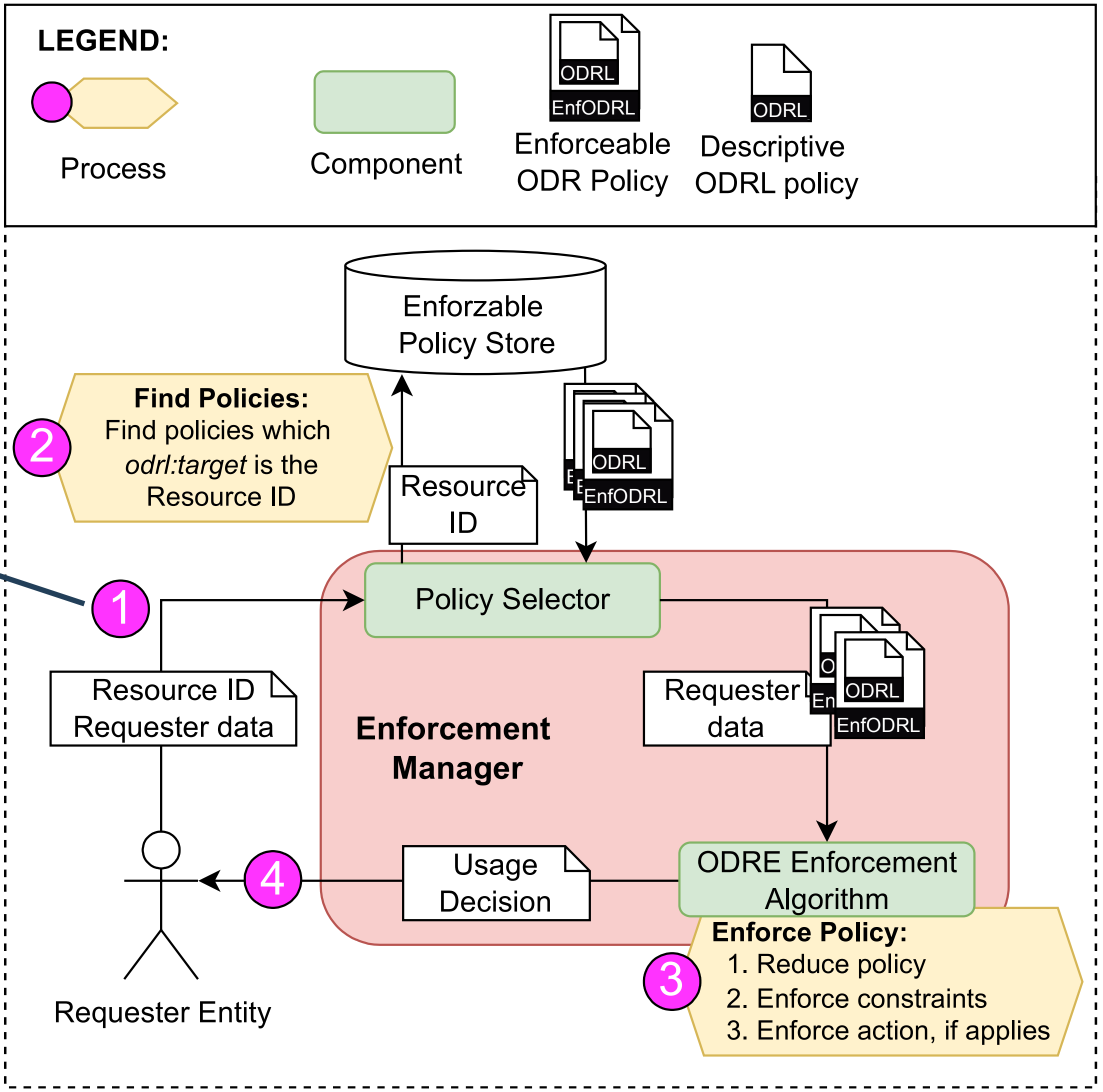
5. **Return the Decision**

Enforcement process



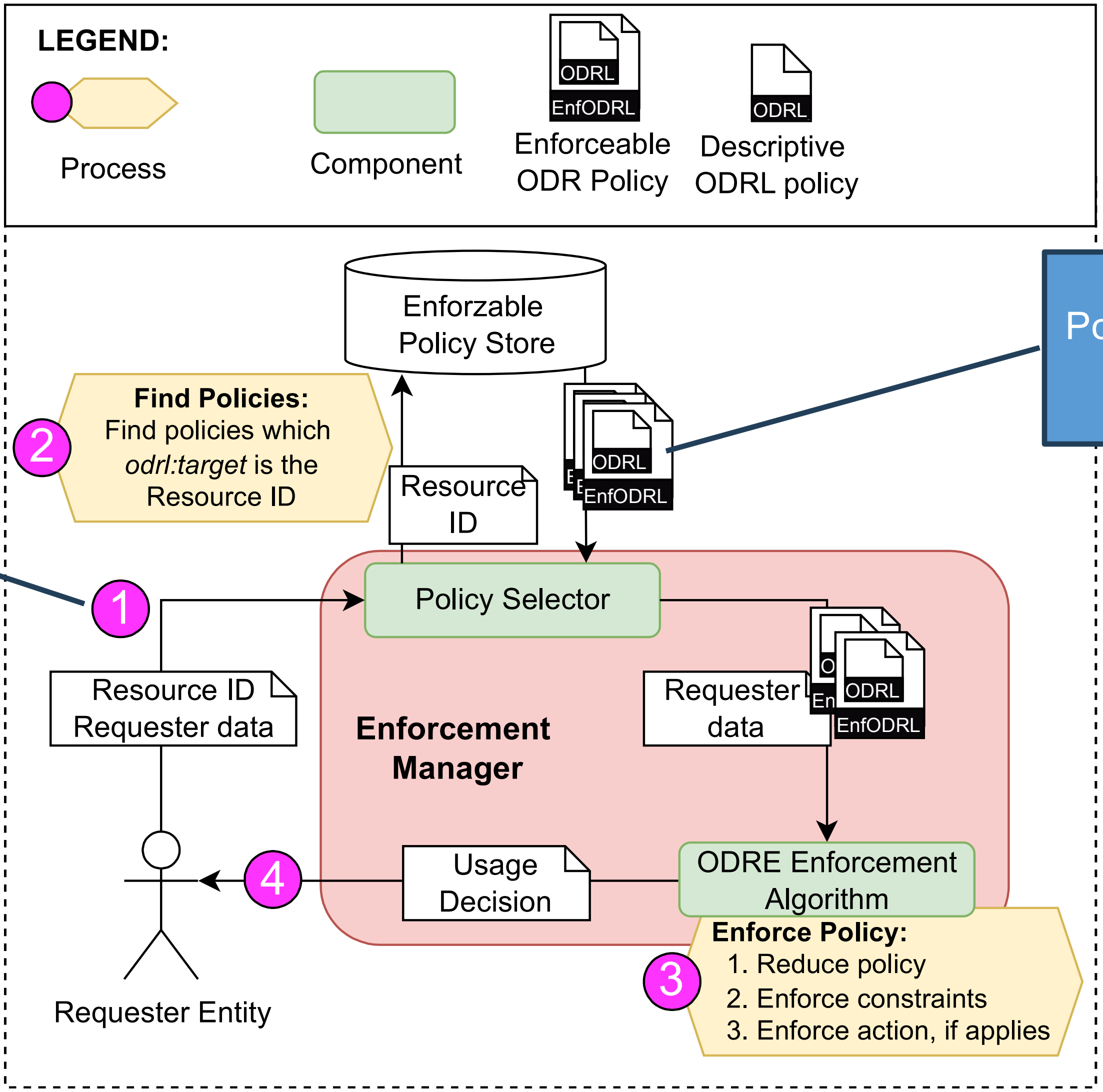
Enforcement process

Provides the ID of the resource is trying to access and, optionally, data needed by the policy



Enforcement process

Provides the ID of the resource is trying to access and, optionally, data needed by the policy



Policies protecting the ID are retrieved

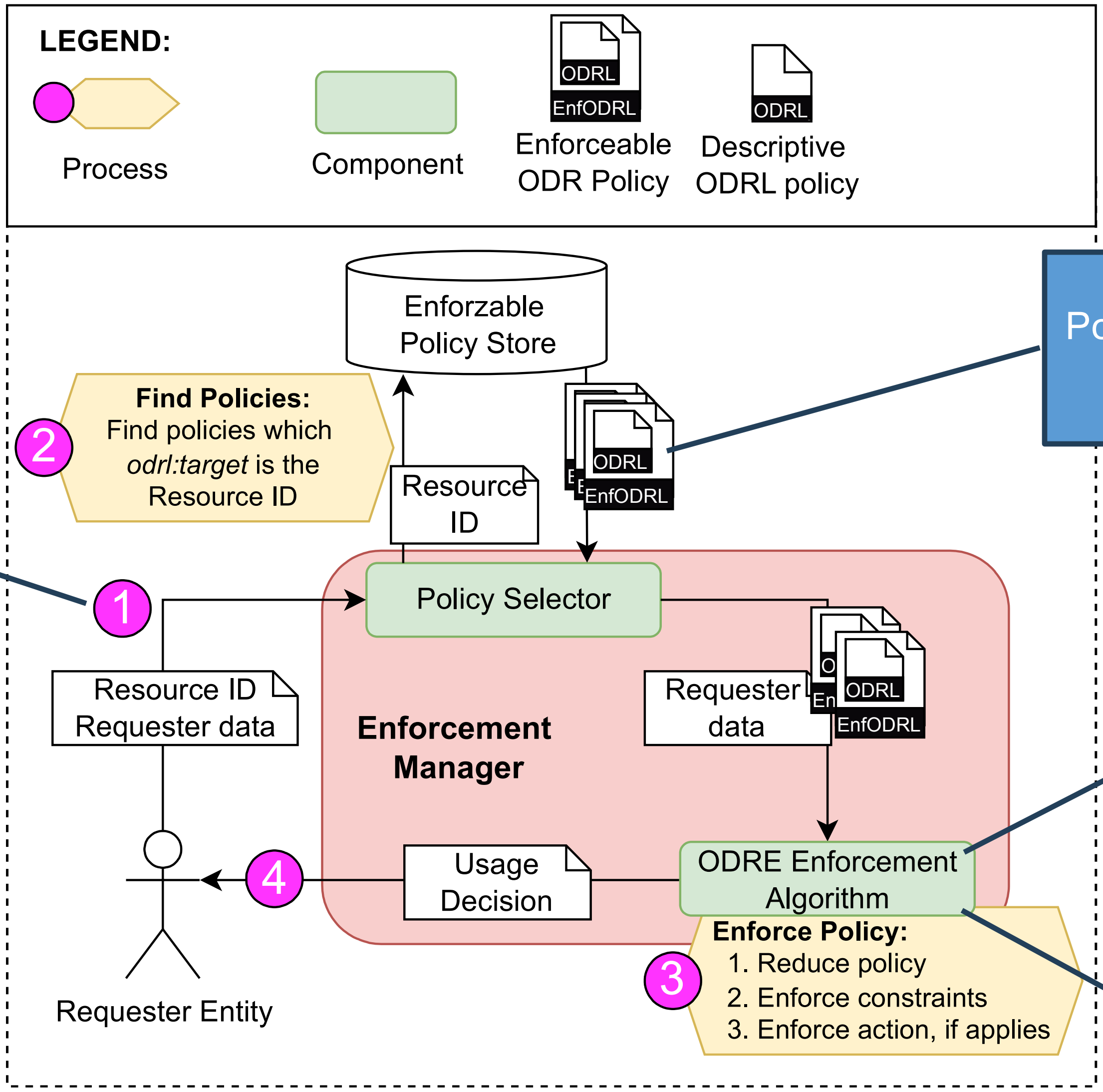
Enforcement process

Provides the ID of the resource is trying to access and, optionally, data needed by the policy

Policies protecting the ID are retrieved

During this process Requester data is injected, and data needed from external services is injected also in the policies before the enforcement

Enforceable actions are performed



Take away

- **ODRE approach aims to be implementation agnostic**
- Its goal is **not to impose a solution** but rather to **define and model the problem and provide an approach to address it**.
- **Multiple improvements exist**:
  - The current limitations where not overcome since this would move away ODRE from the standard.

ODRE approach to enforcement: fine-grinding the ODRL enforcement

Enforcement understanding

Ontology must define the concepts (circumstances, actions, resources, …) unambiguously for people and machines. Lexical differences must no affect the common understanding of policies, ontological terms must be clearly defined so their interpretation is closed.

Requirements from use cases

Use cases must be collected to derive requirements, problems, and practical challenges. These must be addressed by the understanding layer or the theoretical enforcement layer. *For instance, are these common or individual requirements? Can these road-blocks be addressed with technological means or only manually?*

Theoretical enforcement

Should promote an implementation-agnostic proposal; open enough (flexible) to include new practical challenges or use cases and closed enough to be implemented and used

Implementations

Must meet what previous layers defined or established

ODRE implementations

https://github.com/ODRE-Framework/odre-java

https://central.sonatype.com/artifact/io.github.odre-framework/odre-core

https://github.com/ODRE-Framework/odre-python

https://pypi.org/project/odre/

Supported operands: odrl:dateTime, otime:time, otime:date
Supported operators: odrl:eq, odrl:neq, odrl:lt, odrl:lteq, odrl:gt, odrl:gteq, otime:between
Supported actions: demo:dummy_read

DEMO Link

~~ODRE~~ implementations

Many thanks