PIDSKG Workshop (2)

A Policy Framework for Usage Control

Inès Akaichi, Giorgos Flouris, Irini Fundulaki, and Sabrina Kirrane
Disclaimer!

This work is in progress
Usage Control

Context

- An extension of access control
- Regulates usage of the data: permissions (prohibitions) and obligations (dispensations)
- Ensures data sovereignty
- It involves data consumers and data providers/owners
- Related to data storage, distribution, aggregation and processing
- Context of intellectual property protection, privacy protection, compliance with regulations and digital rights management

We focus on policy-based usage control, where we use machine-readable policies to express requirements for future data usage and mechanisms to enforce the respective usage policies.
Usage Control

Context

Figure taken from Usage Control in the International Data Spaces V3.0 (2021). Steinbuss et al.
Usage Control Policy Languages
Related Work

- Usage control policy frameworks/ languages
  - The Obligation Specification Language (OSL) [4]
  - ...

- General policy languages
  - Kaos [5]
  - Rei [6]
  - ...

- Tailored policy languages
  - ODRL [7]
  - The Special Policy Language [8]
  - ...
Usage Control Policy Languages

Gaps

- Usage control policy frameworks/ languages
  - Abstract models [1]
  - Express only obligations [4]
  - Proposed for specific domains [cf., (9)]
  - Lack formal semantics [cf., (1,)]

- General policy languages
  - Not clear how to support general structures encountered in usage control (obligations, dispensations, usage conditions, etc.) [cf., (5,6)]
  - Lack formal semantics [cf., (6)]

- Tailored policy languages
  - Too specific [cf., (7,8)]
  - Lack formal semantics [cf., (7)]
A Language for Usage Control

Requirements

Use Case (1)
The address registration process in Austria
Use Case (2)

Legal requirements

- The legal requirements regarding the registration process in Austria:

  - **Rule 1.** A person is obliged to register their address with one of the local authorities within three days of changing residence or having moved from abroad to Austria.
  - **Rule 2.** A person is obliged to deregister their old address within three days of changing their place of residence, or of leaving the country.
  - **Rule 3.** Tourists in Austria are exempt from registering their address.
  - **Rule 4.** If the person stays in a hotel, they are allowed to request a signature from the hotel.
  - **Rule 5.** If the person stays in with friends or family members, they are allowed to request a signature from the property owner.
  - **Rule 6.** A person is not allowed to open a bank account if they do not have a certificate of registration.
The Usage Control Policy (UCP) Framework

✓ Specification of usage control policies
✓ Representation of the state of affairs via Knowledge Bases
✓ Reasoning tasks
The UCP Framework
Usage Control Policies

Meta-policy

Usage Control Policy 1
Usage Control Policy ...
Usage Control Policy n
Notation

- \( O, D, P, A \) denote the deontic operators **Obligation**, **Dispensation**, **Prohibition**, and **Permission** (allowance)
- \( U \) and \( L \) denote the set of **URIs** and **literals** respectively.
- We also consider two sets, \( P, A \) (subsets of \( U \)), such that \( P \subseteq U, A \subseteq U \)
- Assume additionally the existence of an infinite set \( V \) of variables disjoint from the above sets. We use “\(?\)” to denote variables (e.g., \(?x, ?y\) etc.)
The UCP Framework
Basic Elements: element Pattern

Definition 2 (Element Pattern). An element pattern is a 5-tuple of the form
\((s, pa, o, mp, mo)\) such that:

- \(s \in U \cup V\)
- \(mp \in U \cup V \cup \{\bot\}\)
- \(o \in U \cup L \cup V\)
- \(mo \in U \cup L \cup V \cup \{\bot\}\)
- \(pa \in P \cup A \cup V\)

We denote by \(\mathcal{EP}\) the set of all element patterns.

Example:

Rule 1. A person is obliged to register their address with one of the local authorities within three days of changing residence or having moved from abroad to Austria.

- This rule states that it is an obligation to: \((?x, :\text{register}, ?y, \ldots, \ldots)\)
- Whenever these conditions are true: \((?x, :\text{type}, :\text{Person})\)
  \((?x, :\text{moveTo}, ?y)\)
  \((?y, :\text{type}, :\text{Address})\)
Definition 3 (Deontic Patterns). Let $D = \{O, D, P, A\}$ denote the deontic operators Obligation, Dispensation, Prohibition, and permission (Allowance), respectively. A deontic pattern is a statement of the form $da$, where $d \in D$ and $a \in \mathcal{E}$. 

Example:

Rule 1. A person is *obliged* to *register* their *address* with one of the local authorities *within three days* of changing residence or having moved from abroad to Austria.

$O(?x, :register, ?y, ..., ...)$
The UCP Framework
Usage Control Policies

- A set of rules
- Each rule follows the form:
  - If condition then Aa, Pa, Oa, Da
- Condition: graph pattern (defined based on element patterns)
- Aa, Pa Oa, Da: deontic pattern

Example:

Rule 1. A person is obliged to register their address with one of the local authorities within three days of changing residence or having moved from abroad to Austria.

The UCP Framework

Knowledge Bases
Definition 1 (Element). An element is a 5-tuple of the form \((s, pa, o, mp, mo)\) such that:

- \(s \in U\)
- \(mp \in U \cup \{\perp\}\)
- \(o \in U \cup L\)
- \(mo \in U \cup L \cup \{\perp\}\)
- \(pa \in P \cup A\)

An element \((s, pa, o, mp, mo)\) is called an action element (or simply action) when \(pa \in A\); it is called a factual element (or simply fact) when \(pa \in P\). We denote by \(A\) the set of all actions and by \(F\) the set of all facts.

Example (instantiation):

**Action Element:** (:alice, :register, :boulevard18, :at, :21-08-2022)

**Factual Element:** (:alice, :type, :Person); (:alice, :moveTo, :boulevard18, :at, :22-08-2022); (:boulevard18, :type, :Address)
The UCP Framework
Knowledge Bases

- Factual elements
- Action elements (executed)

Example:

(:alice, :moveTo, :boulevard18, :at, :21-08-2022)
(:alice, :type, :Person)
(:boulevard18, :type, :Address)
(:alice, :register, :boulevard18, :at, :22-08-2022)
Given an element pattern \( a \), modality conflicts arise when:

- \( Oa \) and \( Pa \): both obligated and prohibited from doing \( a \)
- \( Aa \) and \( Pa \): both permitted and prohibited from doing \( a \)
- \( Oa \) and \( Da \): both obligated and exempt from doing \( a \)
The UCP Framework

Usage Control meta-Policies

- Rules
- Precedence relationship $\leq$: order between rules
- Conflict resolution
Precedence Strategies

- Negative policies override positive ones
- Specific overrides general
- New law overrides old law
- Etc.

Example (Dispensation overrides Obligation):
For any two rules $r_1 = \text{cond}_1 \rightsquigarrow \text{Da}_1$, $r_2 = \text{cond}_2 \rightsquigarrow \text{Oa}_2$, such that $a_1 = a_2$, it holds that $r_2 \leq r_1$. 
The UCP Framework
Overview

Usage Control Policies

- Meta-policy
  - Usage Control Policy 1
  - Usage Control Policy ...
  - Usage Control Policy n

Reasoning Tasks

- Consistency
- Normalization
- Compliance
- Policy Querying

Knowledge Bases
Example:

**Rule 1.** A person is obliged to register their address with one of the local authorities within three days of changing residence or having moved from abroad to Austria:

\[
\text{Rule 1. } (?x, \text{moveTo, } ?y). (?x, \text{type, } \text{Person}). (?y, \text{type, } \text{Address}) \quad \text{O}(?x, \text{register, } ?y, \ldots, \ldots)
\]

**Rule 1’.** (?x, :moveTo, ?y). (?x, :type, :Person). (?y, :type, :Address) \quad \text{D}(?x, \text{register, } ?y, \ldots, \ldots)

Inconsistent Policy
Example:

**Rule 1.** A person is obliged to register their address with one of the local authorities within three days of changing residence or having moved from abroad to Austria:

\[
\text{Rule 1. } (?x, :moveTo, ?y). (?x, :type, :Person). (?y, :type, :Address) \quad \leadsto \quad \text{O}(?x, :register, ?y, ..., ...)
\]

\[
\text{Rule 1'. } (?x, :moveTo, ?y). (?x, :type, :Person). (?y, :type, :Address) \quad \leadsto \quad \text{D}(?x, :register, ?y, ..., ...)
\]

Given that Dispensation overrides Obligation: **Rule 1’ would override Rule 1**
The new normalized policy would retain only Rule 1’
Example:

**Rule 1.** A person is obliged to register their address with one of the local authorities within three days of changing residence or having moved from abroad to Austria.

**Policy rule**

\((?x, \text{:moveTo, } ?y). (?x, \text{:type, } \text{:Person}). (?y, \text{:type, } \text{:Address}) \implies \text{O}(?x, \text{:register, } ?y, \text{:within, } \text{:threeDays})\)

**KB**

\((:alice, \text{:moveTo, } \text{boulevard18, } \text{at, } 21-08-2022)\)

\((:boulevard18, \text{:type, } \text{:Address})\)

\((:alice, \text{:type, } \text{:Person})\)

\((:alice, \text{:register, } \text{boulevard18, } \text{at, } 22-08-2022)\)

Compliant or not?
Example:

Given **Rule 1.** a person is obliged to register their address with one of the local authorities within three days of changing residence or having moved from abroad to Austria.

*Question: I would like to move to Austria, what are my obligations?*
Future Steps
Instantiation

- Different Initiatives:
  - ODRL (Ontology Engineering Group at Universidad Politécnica de Madrid)
  - SHACL (L3S research center at Leibniz Universität Hannover)
  - RDF surfaces (IDLab at Ghent University)
  - Description Logics (us)

- Why Description Logics?
  - Decidability
  - Use off-the-shelf reasoners (e.g., FaCT++, HermiT)
Questions

- Use cases
  - So far, the address registration process
  - Others: intellectual property protection, privacy protection, compliance with regulations and digital rights management

- Language representation (DLs, ODRL, etc.)

- Benchmark (The SPECIAL benchmark\(^1\))

---

References


