Semantic Reasoners and Global Restrictions in OWL 2 DL

Valentina Bartalesi Lenzi

Istituto di Scienza e Tecnologie dell'Informazione - Consiglio Nazionale delle Ricerche – Pisa

A.A. 2024-2025

Inference Problems for an OWL DL Ontology

An OWL 2 ontology O is satisfied in an interpretation I if all axioms in the axiom closure of O are satisfied in I

Definition (Model of an OWL Ontology)

Given a datatype map D, an interpretation $I = (\Delta_I, \Delta_D, \cdot^C, \cdot^{OP}, \cdot^{DP}, \cdot^I, \cdot^{DT}, \cdot^{LT}, \cdot^{FA}, \text{NAMED})$ for D is a model of an OWL 2 ontology O w.r.t. D if an interpretation $J = (\Delta_I, \Delta_D, \cdot^C, \cdot^{OP}, \cdot^{DP}, \cdot^J, \cdot^{DT}, \cdot^{LT}, \cdot^{FA}, \text{NAMED})$ for D exists such that $\cdot^J$ coincides with $\cdot^I$ on all named individuals and J satisfies O.

Thus, **an interpretation I satisfying O is also a model of O**.
In contrast, a model I of O may not satisfy O directly; however, by modifying the interpretation of anonymous individuals, we can always be coerced into an interpretation J that satisfies O

Inference Problems

Let D be a datatype map and V a vocabulary over D. Let O and O1 be OWL 2 ontologies, CE, CE1, and CE2 class expressions, and *a* a named individual, such that all of them refer only to the vocabulary V

Ontology Consistency: O is consistent (or satisfiable) w.r.t. D if a model of O w.r.t. D and V exists

Ontology Entailment: O entails O1 w.r.t. D if every model of O w.r.t. D and V is also a model of O1 w.r.t. D and V

Ontology Equivalence: O and O1 are equivalent w.r.t. D if O entails O1 w.r.t. D and O1 entails O w.r.t. D

Inference Problems

Class Expression Satisfiability: CE is satisfiable w.r.t. O and D if a model I of O w.r.t. D and V exists such that $(CE)^C \neq \emptyset$

Class Expression Subsumption: CE1 is subsumed by a class expression CE2 w.r.t. O and D if $(CE1)^C \subseteq (CE2)^C$ for each model I of O w.r.t. D and V

Instance Checking: *a* is an instance of CE w.r.t. O and D if $(a)^I \in (CE)^C$ for each model I of O w.r.t. D and V

Inference Engines

In the field of Artificial Intelligence, an **inference engine** is a component of the system that applies logical rules to the knowledge base to deduce new information

The knowledge base stores facts about the world. The inference engine applies logical rules to the knowledge base and deduces **new knowledge**

Inference Engines

Inference engines are sometimes referred to as **reasoners**

However, basically, **a reasoner and an inference engine are the same thing**

Another name for a reasoner or inference engine is **semantic reasoner**

Inference Engines

A reasoner is a software that is able to **infer logical consequences** from a set of **asserted facts**

Every reasoner uses some sort of **logic**

Every reasoner works with some set of axioms. An axiom describes some logical fact.

The **capabilities of a reasoner** depend on the **expressiveness** of the kind of **logic** that the reasoner uses and the axioms provided for the reasoner.

OWL 2 DL Reasoners

- **HermiT**, hosted at University of Oxford, published under LGPL. HermiT is an OWL 2 DL reasoner.
  Supported reasoning services: classification, satisfiability, entailment, consistency.

- **Chainsaw**, hosted at the University of Manchester, published under LGPL. Chainsaw is a free (LGPL) OWL 2 DL reasoner for very large ontologies. It uses a modular decomposition to tackle the high complexity of the reasoning. Uses delegate reasoner(s) to perform single reasoning tasks.
  Supported reasoning services: classification, satisfiability, entailment, consistency.

- **FaCT++**, hosted at the University of Manchester, published under LGPL. FaCT++ is a free (LGPL) highly optimised open-source C++-based tableaux reasoner for OWL 2 DL.
  Supported reasoning services: classification, satisfiability, entailment, consistency.

- **Pellet** is the OWL 2 DL reasoner open source (AGPL) or commercial license, pure Java, developed and commercially supported by Complexible Inc. Pellet provides functionality to check the consistency of ontologies, compute the classification hierarchy, explain inferences, and answer SPARQL queries.

Open World Assumption

The **Open World Assumption** (OWA) is the assumption that:

what is not known to be true or false might be true,
or
absence of information is interpreted as unknown information, not as negative information.

OWA assumes **incomplete information** about a given state of affairs, i.e., there may be more relevant information than what is provided.

This is useful for describing knowledge in a way that is **extensible** and most commonly used in Artificial Intelligence and throughout the life sciences. This is contrasted with the Closed World Assumption.

Open World Assumption

Take the sample data in the following table and a query: “Which alumni do not have a PhD?”

Alumnus	Degree obtained
Delani	PhD in Molecular Biology
Anna	PhD in Ecology
Peter	MSc in Informatics
Dalila	PhD in Genetics

Then under the OWA, it cannot answer with “Peter” because it does not know if Peter also obtained a PhD: Peter might have, but that has not been represented in the information system yet

To retrieve “Peter” as an answer to the above query, an axiom has to be added that states explicitly that Peter does not have a PhD

Global Restrictions and Axiom Closure

The axiom closure Ax of each OWL 2 DL ontology O **must satisfy global restrictions** that are necessary to obtain a **decidable language**

The formal definition of these conditions relies on the notion of **simple and composite object property expressions** and **property hierarchy**

Composite Object Property Expression

The axiom closure Ax is the smallest set containing OP and INV(OP) for each object property OP occurring in Ax

Definition (Composite Object Property Expression)

An object property expression OPE is *composite in the set of axioms Ax* if OPE is equal to owl:topObjectProperty or owl:bottomObjectProperty, or Ax contains an axiom of the form

- SubObjectPropertyOf(ObjectPropertyChain($OPE_1 \ldots OPE_n$) OPE)
- SubObjectPropertyOf(ObjectPropertyChain($OPE_1 \ldots OPE_n$) INV(OPE))
- TransitiveObjectProperty(OPE)
- TransitiveObjectProperty(INV(OPE)).

Composite Object Property Expression

Definition (Composite Object Property Expression)

An object property expression OPE is *composite in the set of axioms Ax* if OPE is equal to owl:topObjectProperty or owl:bottomObjectProperty, or Ax contains an axiom of the form

- SubObjectPropertyOf(ObjectPropertyChain(OPE_1 ... OPE_n) OPE)
- SubObjectPropertyOf(ObjectPropertyChain(OPE_1 ... OPE_n) INV(OPE))
- TransitiveObjectProperty(OPE)
- TransitiveObjectProperty(INV(OPE)).

Recall: TransitiveObjectProperty(OPE) abbreviates
SubObjectPropertyOf(ObjectPropertyChain(OPE OPE) OPE)

Overall, OPE is **composite** if it, or its inverse, is the **right-hand side** of a **Complex Role Inclusion Axiom (including Transitive object properties!)**, or it is **one of the two built-in object properties** (i.e. owl:topObjectProperty and owl:bottomObjectProperty)

Complex Role Inclusion Axiom

Construct Name	Complex Role Inclusion Axiom
Construct Type	Axiom
Functional Syntax	SubObjectPropertyOf(ObjectPropertyChain($\mathrm{OPE}_1 \ldots \mathrm{OPE}_n$) OPE)
Description	the composition of object property expressions $\mathrm{OPE}_1, \ldots, \mathrm{OPE}_n$ is a sub-property of object property expression OPE
Semantics	$\forall y_0, y_1, \ldots, y_n : (y_0, y_1) \in (OPE_1)^{OP}$ and ... and $(y_{n-1}, y_n) \in (OPE_n)^{OP}$ imply $(y_0, y_n) \in (OPE)^{OP}$
RDF Turtle Syntax	OPE owl:propertyChainAxiom ($\mathrm{OPE}_1 \ldots \mathrm{OPE}_n$) .
Example	

Property Hierarchy Relation and Simple OPEs

The relation → is the smallest relation on AllOPE(Ax) for which the following conditions hold (A → B means that → holds for A and B):

- SubObjectPropertyOf(OPE1 OPE2) ∈ Ax, then OPE1 → OPE2
- EquivalentObjectProperties(OPE1 OPE2) ∈ Ax, then OPE1 → OPE2 and OPE2 → OPE1
- InverseObjectProperties(OPE1 OPE2) ∈ Ax, then OPE1 → INV(OPE2) and INV(OPE2) → OPE1
- SymmetricObjectProperty(OPE) ∈ Ax, then OPE → INV(OPE)
- if OPE1 → OPE2, then INV(OPE1) → INV(OPE2)

Therefore the → relation captures the *explicit* sub-property relationships between object property expressions, *i.e.*, object properties and their inverses.

Simple OPEs

The **property hierarchy** relation $\overset{*}{\rightarrow}$ is the **reflexive-transitive closure** of the sub-property relation →.

Since the **sub-property relation is reflexive and transitive**, $\overset{*}{\rightarrow}$ **includes all consequences of the explicit sub-property relationships in AllOPE(Ax)**

An object property expression **OPE** is ***simple*** in Ax if, for each object property expression OPE' such that **OPE'** $\overset{*}{\rightarrow}$ **OPE, OPE' is not composite**

Examples

Consider the ontology consisting of the following axioms:

SubObjectPropertyOf(ObjectPropertyChain(hasFather hasBrother) hasUncle)
The brother of someone's father is that person's uncle

SubObjectPropertyOf(hasUncle hasRelative)
Having an uncle implies having a relative

SubObjectPropertyOf(hasBiologicalFather hasFather)
Having a biological father implies having a father

- **hasUncle is not simple**, because it occurs in the right-hand side of a complex role inclusion axiom
- **hasRelative is not simple** either, because it has a subproperty that is not simple
- **hasBiologicalFather is simple**, and so is hasFather

Global Restrictions

The set of axioms Ax satisfies the global restrictions of OWL 2 DL if it satisfies all the restrictions that are introduced next.

These restrictions concern:

1. **owl:topDataProperty**
2. **Datatypes**
3. **Simple roles**
4. **Property hierarchy**
5. **Anonymous individuals**

Restriction on owl:topDataProperty

The owl:topDataProperty property occurs in Ax only as the second argument of SubDataPropertyOf axioms

In practice, this restriction **prohibits defining super-properties of owl:topDataProperty**.
Without this restriction, owl:topDataProperty could be used to write axioms about datatypes, which would invalidate the following theorem:

Theorem (DS1: Independence of Direct Semantics from the Datatype Map)

Let O_1 and O_2 be OWL 2 DL ontologies over a vocabulary V and $D = (NDT, NLS, NFS, \cdot^{DT}, \cdot^{LS}, \cdot^{FS})$ a datatype map such that each datatype mentioned in O_1 and O_2 is rdfs:Literal, a datatype defined in the respective ontology, or it occurs in NDT. Furthermore, let $D' = (NDT', NLS', NFS', \cdot^{DT'}, \cdot^{LS'}, \cdot^{FS'})$ be a datatype map such that $NDT \subseteq NDT'$, $NLS(DT) = NLS'(DT)$, and $NFS(DT) = NFS'(DT)$ for each $DT \in NDT$, and $\cdot^{DT'}$, $\cdot^{LS'}$ and $\cdot^{FS'}$ are extensions of $\cdot^{DT}$, $\cdot^{LS}$ and $\cdot^{FS}$, respectively. Then, O_1 entails O_2 w.r.t. D if and only if O_1 entails O_2 w.r.t. D'.

Restriction on owl:topDataProperty

This restriction guarantees that **adding datatypes to a datatype map does not alter entailment**, and it has several useful consequences:

- One can apply the direct semantics to an OWL 2 DL ontology O by considering **only the datatypes explicitly occurring in O**
- When referring to various reasoning problems, the datatype map D need not be given explicitly, as it is sufficient to consider an implicit **datatype map containing only the datatypes from the given ontology**
- OWL 2 DL reasoners can provide datatypes not explicitly mentioned in this specification **without fear that this will change the meaning of OWL 2 DL ontologies** not using these datatypes

Restriction on owl:topDataProperty

However, if the **restriction is violated**, the theorem no longer holds, and the **consequences** of an ontology would then not necessarily depend **only on the datatypes used in the ontology**, **but would also depend on the datatypes selected in the OWL 2 datatype map**

Thus, if an implementation or a future revision of OWL decided to extend the set of supported datatypes, it would run **the risk of possibly changing the consequences of certain ontologies**

Restriction on Datatypes

1) Each datatype occurring in Ax satisfies exactly one of the following conditions: it is rdfs:Literal, or it is contained in the OWL 2 datatype map, or it is defined by a single datatype definition axiom in Ax

2) A strict partial order (*i.e.*, an irreflexive and transitive relation) < on the set of all datatypes in Ax exists such that, for each axiom of the form DatatypeDefinition(DT DR) and each datatype DT1 occurring in DR, we have DT1 < DT

The first condition ensures that all datatypes in Ax are given a **well-defined interpretation** and that datatype definitions do not redefine the datatypes from the OWL 2 datatype map

The second condition ensures that datatype **definitions are acyclic**, that is, if a datatype DT1 is used in a definition of DT, then DT is not allowed to be used in the definition of DT1

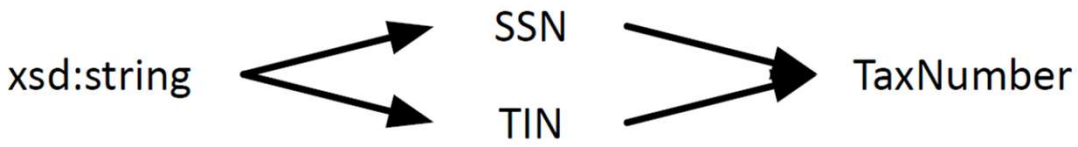
Restriction on Datatypes

Example:

```
Declaration( Datatype( SSN ) )
Declaration( Datatype( TIN ) )
Declaration( Datatype( TaxNumber ) )
DatatypeDefinition(
SSN
DatatypeRestriction( xsd:string xsd:pattern "[0-9]3-[0-9]2-[0-9]4" ) )
DatatypeDefinition(
TIN
DatatypeRestriction( xsd:string xsd:pattern "[0-9]11" ) )
DatatypeDefinition( TaxNumber DataUnionOf( SSN TIN ) )
```

Restriction on Datatypes

These datatype definitions are acyclic: SSN and TIN are defined in terms of xsd:string, and TaxNumber is defined in terms of SSN and TIN



To verify this condition formally, it suffices to find one strict partial order < on these datatypes such that each datatype is defined only in terms of the datatypes that are smaller w.r.t. <

xsd:string < SSN < TaxNumber
xsd:string < TIN < TaxNumber

Restriction on Simple Roles

Each class expression and each axiom in Ax of type from the following two lists contains only ***simple*** **object properties**:

(1) ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, and ObjectHasSelf

(2) FunctionalObjectProperty, InverseFunctionalObjectProperty, IrreflexiveObjectProperty, AsymmetricObjectProperty, and DisjointObjectProperties

This restriction is necessary in order to guarantee **decidability** of the basic reasoning problems for OWL 2 DL

Restriction on the Property Hierarchy

A **strict partial order** (*i.e.*, an irreflexive and transitive relation) < on AllOPE(Ax) exists that fulfills the following conditions:

1) OPE1 < OPE2 if and only if INV(OPE1) < OPE2 for all object properties OPE1 and OPE2 occurring in AllOPE(Ax)
2) If OPE1 < OPE2 holds, then OPE2 → OPE1 does not hold;
3) Each axiom in Ax of the form
SubObjectPropertyOf(ObjectPropertyChain(OPE1 . . . OPE*n*) OPE)
with $n \geq 2$ fulfills the following conditions:

- 3.a) OPE is equal to owl:topObjectProperty, or
- 3.b) $n = 2$ and OPE1 = OPE2 = OPE, or
- 3.c) OPEi < OPE for each $1 \leq i \leq n$, or
- 3.d) OPE1 = OPE and OPEi < OPE for each $2 \leq i \leq n$, or
- 3.e) OPEn = OPE and OPEi < OPE for each $1 \leq i \leq n - 1$

This restriction is necessary in order to **guarantee decidability** of the basic reasoning problems for OWL 2 DL

Restriction on the Property Hierarchy

The main goal of this restriction is to prevent **cyclic definitions involving object sub property axioms with property chains**. Consider the following ontology:

SubObjectPropertyOf(ObjectPropertyChain(hasFather hasBrother) hasUncle)
The brother of someone's father is that person's uncle

SubObjectPropertyOf(ObjectPropertyChain(hasUncle hasWife) hasAuntInLaw)
The wife of someone's uncle is that person's aunt-in-law

The second axiom depends on the first one, but not vice versa; hence, these axioms are **not cyclic** and can occur together in the axiom closure of an OWL 2 DL ontology

Restriction on the Property Hierarchy

To verify this condition formally, it suffices to find one strict partial order < on object properties such that each property is defined only in terms of the properties that are smaller w.r.t. <.
For example:

hasFather	hasUncle
hasBrother	hasUncle
hasUncle	hasAuntInLaw
hasWife	hasAuntInLaw

Restriction on the Property Hierarchy

In contrast to the previous example, the following axioms are cyclic and do not satisfy the restriction on the property hierarchy:

SubObjectPropertyOf(ObjectPropertyChain(hasFather hasBrother) hasUncle)
The brother of someone's father is that person's uncle

SubObjectPropertyOf(ObjectPropertyChain(hasChild hasUncle) hasBrother)
The uncle of someone's child is that person's brother

The second axiom depends on the first one and vice versa; hence, these axioms are **cyclic** and **cannot occur together** in the axiom closure of an OWL 2 DL ontology

Restrictions on Anonymous Individuals

1) No anonymous individual occurs in Ax in an axiom of type **SameIndividual, DifferentIndividuals, NegativeObjectPropertyAssertion, or NegativeDataPropertyAssertion**

2) No anonymous individual occurs in Ax in a class expression of type **ObjectOneOf** or **ObjectHasValue**

3) The anonymous individual graph for Ax is the **undirected graph** F whose vertices are anonymous individuals occurring in Ax, and that contains an (undirected) edge between each pair of anonymous individuals _:x and _:y for each assertion in Ax of the form ObjectPropertyAssertion(OPE _:x _:y)

Restrictions on Anonymous Individuals

These restrictions ensure that each OWL 2 DL ontology with anonymous individuals can be transformed to an equivalent **ontology without anonymous individuals**, thereby ensuring **decidability** of the basic reasoning problems

Global Restrictions in Practice

Consider the following axioms to formally represent the domain related to scientific publications:

1. Each journal is published by exactly one publisher.
2. Each publisher uses exactly one open access repository.
3. Each article can be a journal article or a conference article.
4. Each journal volume is an issue of exactly one journal and has exactly one number and one date.
5. A journal article is collected in at least one repository.
6. An article that is part of a journal volume that is an issue of a journal published by a publisher that uses an open access repository is collected in that repository.

Example

1. Restriction on owl:topDataProperty → This is satisfied because the ontology does not include any axiom on owl:topDataProperty and any superproperty of topDataProperty was defined
2. Restrictions on Datatypes → This is satisfied because the ontology:
   - only uses datatypes from the OWL 2 datatype map
   - does not define any data range
3. Restriction on Simple Roles → This is satisfied because no composite object property is used in an axiom of the forbidden types
4. Restriction on the Property Hierarchy → This is satisfied because there is only one property chain in the ontology and it does not introduce cycles
5. Restrictions on Anonymous Individuals → This is satisfied because there are no anonymous individuals in our ontology

Conclusions

We have examined OWL 2 DL, the most recent language for knowledge representation on the web

OWL DL 2 is based on the description logic SROIQ, which is the most expressive decidable description logic that offers a level of expressivity adequate to realistic applications

In order to retain decidability, several restrictions are placed on an OWL2 DL ontology. These restrictions limit the expressivity of the language, but at the same time, they guarantee full control of the systems built on top of OWL 2 DL

The engineering of inference engines able to reason about OWL 2 DL ontologies is a current research topic

Useful Readings

- Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, eds. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition) W3C Recommendation, 11 December 2012. http://www.w3.org/TR/owl2-syntax/

- Boris Motik, Peter F. Patel-Schneider, Bernardo Cuenca Grau, eds. OWL 2 Web Ontology Language: Direct Semantics (Second Edition) W3C Recommendation, 11 December 2012. http://www.w3.org/TR/owl2-direct-semantics/

- Michael Schneider, editor. OWL 2 Web Ontology Language: RDF-Based Semantics (Second Edition) W3C Recommendation, 11 December 2012. http://www.w3.org/TR/owl2-rdf-based-semantics/