Global Journal of Computer Science and Technology: C
Software & Data Engineering
Volume 18 Issue 2 Version 1.0 Year 2018
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# The Method of Normalizing OWL 2 DL Ontologies

By Malgorzata Sadowska & Zbigniew Huzar

*Wroclaw University of Science and Technology*

*Abstract-* The paper proposes a method of normalizing OWL 2 DL ontologies. The method introduces rules aimed at refactoring OWL 2 constructs. The proposed transformations only use a subset of OWL 2 constructs and enable to present an input OWL 2 ontology in a new but semantically equivalent form. The normalization is motivated by the fact that normalized OWL 2 DL ontologies have a unified structure of axioms so that they can be compared in an algorithmic way.

*Keywords:* *OWL 2, normalization.*

*GJCST-C Classification:* *H.3.5*


*Strictly as per the compliance and regulations of:*

# The Method of Normalizing OWL 2 DL Ontologies

Małgorzata Sadowska [α] & Zbigniew Huzar [σ]

*Abstract -* The paper proposes a method of normalizing OWL 2 DL ontologies. The method introduces rules aimed at refactoring OWL 2 constructs. The proposed transformations only use a subset of OWL 2 constructs and enable to present an input OWL 2 ontology in a new but semantically equivalent form. The normalization is motivated by the fact that normalized OWL 2 DL ontologies have a unified structure of axioms so that they can be compared in an algorithmic way.

*Keywords: OWL 2, normalization.*

## I. Introduction

OWL 2 Web Ontology Language (OWL 2) [1] is a language of knowledge representation used for defining ontologies. The ontologies which satisfy syntactic conditions listed in the specification in Section 3 of [2] are called OWL 2 DL ontologies and have their semantics expressed in *SROIQ* description logic [1]. *SROIQ* was designed to provide additions to OWL-DL to guarantee decidability in reasoning [3, 4].

The domain ontologies are expected to provide a knowledge base about specific application area. Therefore they should be consistent. An ontology consistency check [5] is one of the reasoning problems that can be answered with the use of inference engines.

In our work, we would like to take advantage of and reuse the existing OWL 2 DL domain ontologies. We assume that the selected OWL 2 DL domain ontology is syntactically correct, consistent and adequately describes the notions from the needed domain. One can successfully conduct reasoning over the ontology with the use of one of the reasoning engines available. However, it is not obvious or conclusive how to effectively process other useful operations on ontologies, for example how to compare two or more ontologies. The problem of comparing two ontologies with the agreed vocabulary was faced in [6] in a method of semantic validation of UML class diagrams. In [6], in the beginning, the UML class diagram is transformed into an ontology expressed in OWL 2. Next, the two ontologies-the domain ontology and the ontological representation of the UML diagram-need to be compared against each other.

The question arises: how to correctly and automatically find out whether one ontology is compliant or contradictory concerning another one ? For the purpose of answering the question, we introduce such a form of normalization that allows for unifying the structure of axioms in the ontologies so that it is possible to automatically compare them. The method of normalization of ontologies and the method of semantic validation of UML class diagrams has been implemented in the prototype tool [8]. This paper presents the details of conducting the transformation of OWL 2 ontology to its normalized form. The important fact is that the presented transformations only change the structure but do not affect the semantics of axioms or expressions in the OWL 2 ontology.

We propose the following groups of transformations of OWL 2 constructs:

*Group I. Extraction of declarations of entities:* A declaration in OWL 2 associates an *Entity* with its type. If a declaration axiom for the selected *Entity* is missing from the ontology, it can be retrieved based on the usage of the *Entity*. In OWL 2, the declaration axiom can be specified for all types of entities: *Class*, *Datatype*, *ObjectProperty*, *DataProperty*, *AnnotationProperty* and *NamedIndividual*.

*Group II. Removal of duplicates in data ranges, expressions, and axioms:* Following [2], sets written in one of the exchange syntaxes (e.g., XML or RDF/XML) may contain duplicates. Therefore, duplicates (if applicable) are eliminated from axioms (e.g. *EquivalentClasses*), data ranges (e.g. *DataUnionOf*) and expressions (e.g. *DataUnionOf*).

*Group III. Restructuration of data ranges and expressions:* The proposed restructurations are intended (1) to flatten the nested structures of the data ranges and expressions, (2) to eliminate the weakest cardinality restrictions included in the data ranges or expressions, and (3) to refactor the data ranges and expressions which are connected with union, intersection and complement constructors, based on the rules of the De Morgan's laws.

*Group IV. Removal of syntactic sugar in axioms:* OWL 2 offers the so-called syntactic sugar [4]. The syntactic sugar makes some axioms easier to write and read for humans (e.g., *DisjointUnion* axiom) but does not lend itself so easily to processing conducted by tools. Due to this fact, the removal of syntactic sugar allows e.g., easier comparison of axioms expressing the same semantics but written with a different syntax.

*Group V. Restructuration of axioms:* Most of OWL 2 axioms which contain several class expressions can be

*Author α σ: Faculty of Computer Science and Management, Wroclaw University of Science and Technology.*
*e-mails: m.sadowska, zbigniew.huzar@pwr.edu.pl*

restructured into several axioms containing only two class expressions each, e.g., *DisjointClasses* and *EquivalentClasses* axioms. This is only worth to be applied for axioms whose order of internal expressions is not important.

*Group VI. Removal of duplicated axioms:* A correctly specified OWL 2 ontology cannot contain two identical axioms (see Section 3). However, duplicated axioms may appear as a result of applying transformations from groups IV and V. Therefore, the last step of the normalization algorithm is to remove all duplicate axioms from the output ontology.

We define *ontology normalization* as a process of transforming the input ontology into the ontology in its refactored form. In Section 4, we presented replacing and replaced OWL 2 constructs used in the process of normalizing OWL 2 DL ontologies. The details of the ontology normalization algorithm are presented in Section 5. The process consists of four phases, which are executed in the following order in the algorithm:

1. Extraction of declarations (group I).
2. Refactorization of data ranges and expressions through applying transformations from group II.
3. Restructuration of expressions and data ranges through applying transformations from group III.
4. Refactorization of axioms through applying transformations from groups II, IV, V and VI.

We consider the output ontology (obtained as a result of conducting the algorithm) as *normalized.* Because all transformations (of the replaced OWL 2 constructs to the replacing OWL 2 constructs) preserve semantics, the semantics of the normalized ontology is the same as the semantics of the input ontology.

In this paper, OWL 2 constructs are written with the use of functional-style syntax [2]. Additionally, the following convention is used:

- C – indicates a class,
- CE (possibly with an index) indicates a class expression,
- OP – indicates an object property,
- OPE (possibly with an index)– indicates an object property expression,
- DP – indicates a data property,
- DPE (possibly with an index) indicates a data property expression,
- DR – indicates a data range,
- a – indicates an individual,
- lt – indicates a literal,
- $\alpha = \beta$ – means the textual identity of $\alpha$ and $\beta$ OWL 2 constructs.

## II. Related Work

According to our investigation, a similar concept of normalization of OWL 2 ontologies has not yet been proposed. In this paper, the normalization is aimed at unifying the structure of axioms in the ontologies allowing for automatic processing of the ontologies. A different purpose (as well as a different kind of) ontology normalization has been proposed in [9], where ontology normalization was suggested to be a pre-processing step that aligns structural metrics with intended semantic measures. Additionally, in [10] and [11], normalization has been proposed as an aspect of ontology design that provides support for ontology reuse, maintainability, and evolution. In [10] and [11] the criteria for normalization are focused on engineering issues that make ontologies modular and understandable for knowledge engineers.

## III. OWL 2 Ontology as a Set of Axioms

The structural specification of OWL 2 [2] is defined with the use of Unified Modeling Language (UML) [7], and the notation is compatible with Meta-Object Facility (MOF) [12]. OWL 2 ontologies consist of entities (classes, datatypes, object properties, data properties, annotation properties and named individuals), expressions (class expressions, data and object property expressions) and axioms (e.g., subclass axioms).

The main component of OWL 2 ontologies is axioms (see fig.1) which specify what is true in a specific domain. The expressions are used to represent complex notions in the described domain. Textually, expressions can be seen as components of axioms, for example, two or more class expressions are needed to specify *DisjointClasses* axiom (see fig. 2). Finally, entities constitute the vocabulary of an ontology.

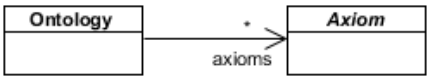


*Fig.1:* A relation between OWL 2 ontology and axioms (extract from Figure 1 of OWL 2 specification [2]).

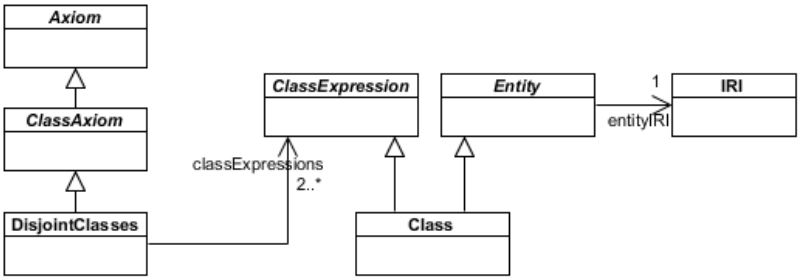


*Fig.2:* A relation between the selected class axiom, relevant expressions and entities (by OWL 2 specification [2]).

Because the association end named *axioms* (see fig. 1) is specified with the use of UML *Multiplicity Element* and a *Set* collection type (*is Ordered*=false and *is Unique*=true), a correct OWL 2 ontology cannot contain two axioms that are textually equivalent. In the normalization method, it is assured through applying the transformations from group VI.

Nevertheless, the ontology may have axioms which contain the same information. For example, it may include the following two axioms: *DisjointUnion* (:*Child* :*Boy* :*Girl* ) and *DisjointClasses*(:*Boy*: *Girl*). The semantics of DisjointUnion [2] states that *Child* class is a disjoint union of *Boy* and *Girl* class expressions which are pairwise disjoint. Therefore, the additional information specified by *DisjointClasses* can be seen as redundant and will be refactored with the proposed transformation rules (here from groups II and IV).

The structural specification of OWL 2 [2] defines an abstract class *Axiom* (see fig. 3). The abstract class *Axiom* is specialized by the following classes: *ClassAxiom* (abstract), *ObjectPropertyAxiom* (abstract), *DataPropertyAxiom* (abstract), *Declaration*, *Datatype Definition* (abstract), *HasKey*, *Assertion* (abstract) and *AnnotationAxiom* (abstract).

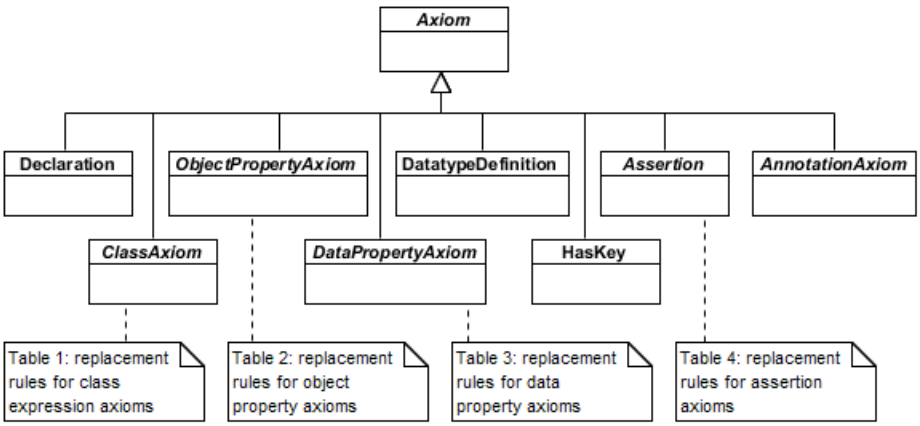


*Fig. 3:* The axioms of OWL 2 [2]and the tables which specify the proposed replacement rules.

*AnnotationAxiom* [2] axioms are mainly used to improve readability for humans. The axioms do not affect the semantics [2]. Therefore, they are not further restructured in this paper.

*Declaration* [2] axioms specify that entities are part of the vocabulary in ontology and are of a specific type, e.g., class, datatype, etc. OWL 2 DL ontology must explicitly declare all datatypes that occur in datatype definition, although in general, it is advisable to declare all entities for verification of the correctness of the usage of the entity based on its type. In the normalization method, if a declaration axiom is missing from the ontology, it is automatically retrieved based on the entity usage (transformation from the group I). This is applied toall types of entities but *AnnotationProperty*, because *AnnotationProperty* is only used to provide annotation and has no effect on the semantics.

*Datatype Definition* [2] axiom defines a new datatype as being semantically equivalent to a unary data range. The *Datatype Definition* axiom is defined in the form that does not need to be restructured. Nonetheless, the data ranges included in other axioms or expressions may require refactoring (transformation from group III). The replacement rules for data ranges are presented in Table 5.

*HasKey*[2] axiom states that each named instance of the specified class expression is uniquely identified by the specified object property and/or data property expressions. It is useful in querying about individuals which are uniquely identified. The *HasKey*

axiom itself is defined in the form that does not need to be restructured, but the class expression and object property expressions included in the axiom are restructured by the transformations presented in Tables 6 and 7 (transformation from group III).

To summarize, Section 4 presents replacement rules for *Class Axioms* in Table 1, for *Object Property Axioms* in Table 2, for *Data Property Axioms* in Table 3 and *Assertion* axioms in Table 4, Table 5 presents replacement rules for data ranges, Table 6 - for class expressions and Table 7 for object property expressions. Each row in Tables 1-7 contains the number of the transformation group (by Groups I-VI defined in Introduction). Additionally, all the transformations from Group III are marked with the sub-number (1)-(3) which defines a concrete refactorization within the group.

## IV. OWL 2 Construct Replacements

Each replaced OWL 2 construct is semantically equivalent to the defined replacing construct(s). Most of the proposed transformations are our original proposals, but some of them come from the OWL 2 specification [2]. The specification defines the so-called syntactic sugar for selected axioms in more detail. This is for ease of writing of some popular patterns for humans. It is cited, where applicable, separately in each table.

*a) Class expression axioms*

OWL 2 class expression axioms define the relationships between class expressions. The abstract class *ClassAxiom* is specified by the following concrete classes: *SubClassOf*, *EquivalentClasses*, *DisjointClasses* and *DisjointUnion*. In Table 1, transformations of IDs: 3, 6 and 8 are defined in [2], the other transformations are proposed by us. The replacing axioms in ID 6 are semantically equivalent.

*Table 1:* Replaced and replacing class expression axioms.

| ID | Group | Replaced axiom | Replacing axiom(s) |
|---|---|---|---|
| 1 | II | EquivalentClasses( $CE_1$ ... $CE_i$ ... $CE_j$ ... $CE_N$) and $1 \le i \le j \le N$ and $N \ge 3$ and $CE_i = CE_j$ | EquivalentClasses( $CE_1$ ... $CE_i$ ... $CE_N$) and $1 \le i \le N$ and $N \ge 2$ |
| 2 | V | EquivalentClasses( $CE_1$ ... $CE_N$) and $1 \le i \le N$ and $N \ge 2$ | EquivalentClasses ( $CE_i$ $CE_j$ ) and $i,j \in \{1,N\}$ and $i \ne j$ and $N \ge 2$ |
| 3 | IV | EquivalentClasses( $CE_1$ $CE_2$ ) | SubClassOf( $CE_1$ $CE_2$ ) SubClassOf( $CE_2$ $CE_1$ ) |
| 4 | II | DisjointClasses( $CE_1$ ... $CE_i$... $CE_j$ ... $CE_N$) and $1 \le i \le j \le N$ and $N \ge 3$ and $CE_i = CE_j$ | DisjointClasses( $CE_1$ ... $CE_i$ ... $CE_N$) and $1 \le i \le N$ and $N \ge 2$ |
| 5 | V | DisjointClasses( $CE_1$ ... $CE_N$ ) and $N \ge 2$ | DisjointClasses( $CE_i$ $CE_j$ ) and $i,j \in \{1,N\}$ and $i \ne j$ and $N \ge 2$ |
| 6 | IV | DisjointClasses( $CE_1$ $CE_2$ ) | SubClassOf ($CE_1$ ObjectComplementOf( $CE_2$ ) ) SubClassOf ($CE_2$ ObjectComplementOf( $CE_1$ ) ) |
| 7 | II | DisjointUnion( C $CE_1$ ... $CE_i$ ... $CE_j$ ... $CE_N$) and $1 \le i \le j \le N$ and $N \ge 3$ and $CE_i = CE_j$ | DisjointUnion( C $CE_1$ ... $CE_i$ ... $CE_N$) and $1 \le i \le N$ and $N \ge 2$ |
| 8 | IV | DisjointUnion( C $CE_1$ ... $CE_N$ ) and $N \ge 2$ | EquivalentClasses( C ObjectUnionOf ( $CE_1$ ... $CE_N$ ) ) DisjointClasses( $CE_1$ ... $CE_N$ ) and $N \ge 2$ |

*b) Object property axioms*

OWL 2 object property axioms define the relationships between property expressions. The abstract class *ObjectPropertyAxiom* is specified by the following concrete classes: *SubObjectPropertyOf*, *EquivalentObjectProperties*, *Disjoint Object Properties*, *InverseObjectProperties*, *ObjectPropertyDomain*, *Object PropertyRange*, *ReflexiveObjectProperty*, *Irreflexive ObjectProperty*, *FunctionalObjectProperty*, *Inverse FunctionalObjectProperty*, *SymmetricObjectProperty*, *AsymmetricObjectProperty* and *TransitiveObjectProperty*. In Table 2, transformations of IDs: 3 and 6-14 are defined in [2], the other transformations are proposed by us. The replacing axioms in ID 6 are semantically equivalent.

*Table 2:* The replaced and replacing object property axioms.

| ID | Group | Replaced axiom | Replacing axiom(s) |
|---|---|---|---|
| 1 | II | EquivalentObjectProperties( $OPE_1$ ... $OPE_i$ ... $OPE_j$ ... $OPE_N$) and 1 ≤ i ≤ j ≤ N and N ≥ 3 and $OPE_i = OPE_j$ | EquivalentObjectProperties( $OPE_1$ ... $OPE_i$ ... $OPE_N$) and 1 ≤ i ≤ N and N ≥ 2 |
| 2 | V | EquivalentObjectProperties( $OPE_1$ ... $OPE_N$ ) and 1 ≤ i ≤ N and N ≥ 2 | EquivalentObjectProperties( $OPE_i$ $OPE_j$ ) and i,j ∈ {1,N} and i ≠ j and N ≥ 2 |
| 3 | IV | EquivalentObjectProperties( $OPE_1$ $OPE_2$ ) | SubObjectPropertyOf( $OPE_1$ $OPE_2$ ) SubObjectPropertyOf( $OPE_2$ $OPE_1$ ) |
| 4 | II | DisjointObjectProperties( $OPE_1$ ... $OPE_i$... $OPE_j$ ... $OPE_N$) and 1 ≤ i ≤ j ≤ N and N ≥ 3 and $OPE_i = OPE_j$ | DisjointObjectProperties( $OPE_1$ ... $OPE_i$ ... $OPE_N$) and 1 ≤ i ≤ N and N ≥ 2 |
| 5 | V | DisjointObjectProperties( $OPE_1$ ... $OPE_N$ ) and 1 ≤ i ≤ N and N ≥ 2 | DisjointObjectProperties( $OPE_i$ $OPE_j$ ) and i,j ∈ {1,N} and i ≠ j and N ≥ 2 |
| 6 | IV | InverseObjectProperties( $OPE_1$ $OPE_2$ ) | EquivalentObjectProperties( $OPE_1$ ObjectInverseOf( $OPE_2$ ) ) EquivalentObjectProperties( $OPE_2$ ObjectInverseOf( $OPE_1$ ) ) |
| 7 | IV | ObjectPropertyDomain( OPE CE ) | SubClassOf (Object Some Values From(OPE owl:Thing ) CE ) |
| 8 | IV | ObjectPropertyRange( OPE CE ) | SubClassOf( owl:Thing ObjectAllValuesFrom( OPE CE ) ) |
| 9 | IV | FunctionalObjectProperty( OPE ) | SubClassOf( owl:Thing ObjectMaxCardinality( 1 OPE ) ) |
| 10 | IV | InverseFunctionalObjectProperty( OPE ) | SubClassOf (owl:Thing ObjectMaxCardinality( 1 ObjectInverseOf( OPE ) ) ) |
| 11 | IV | ReflexiveObjectProperty( OPE ) | SubClassOf( owl:Thing ObjectHasSelf( OPE ) ) |
| 12 | IV | IrreflexiveObjectProperty( OPE ) | SubClassOf( ObjectHasSelf( OPE ) owl:Nothing ) |
| 13 | IV | SymmetricObjectProperty( OPE ) | SubObjectPropertyOf( OPE ObjectInverseOf( OPE ) ) |
| 14 | IV | TransitiveObjectProperty( OPE ) | SubObjectPropertyOf (ObjectPropertyChain( OPE OPE ) OPE ) ) |

c) *Data property axioms*

OWL 2 data property axioms define the relationships between property expressions. The abstract class *DataProperty Axiom* is specified by the following concrete classes: *SubDataProperty Of*, *EquivalentDataProperties*, *DisjointDataProperties*, *Data PropertyDomain*, *DataPropertyRange*, and *Functional DataProperty*.In Table 3, transformations of IDs: 3 and 6-8 are defined in [2], the remaining transformations are proposed by us.

*Table 3:* The replaced and replacing data properties axioms.

| ID | Group | Replaced axiom | Replacing axiom(s) |
|---|---|---|---|
| 1 | II | EquivalentDataProperties ( $DPE_1$ ... $DPE_i$... $DPE_j$ ... $DPE_N$) and 1 ≤ i ≤ j ≤ N andN ≥ 3 and $DPE_i = DPE_j$ | EquivalentDataProperties ( $DPE_1$ ... $DPE_i$ ... $DPE_N$) and 1 ≤ i ≤ N and N ≥ 2 |
| 2 | V | EquivalentDataProperties( $DPE_1$ ... $DPE_N$ ) and 1 ≤ i ≤ N and N ≥ 2 | EquivalentDataProperties( $DPE_i$ $DPE_j$ ) and i,j ∈ {1,N} and i ≠ j and N ≥ 2 |
| 3 | IV | EquivalentDataProperties( $DPE_1$ $DPE_2$ ) | SubDataPropertyOf( $DPE_1$ $DPE_2$ ) SubDataPropertyOf( $DPE_2$ $DPE_1$ ) |
| 4 | II | DisjointDataProperties( $DPE_1$ ... $DPE_i$... $DPE_j$ ... $DPE_N$) and 1 ≤ i ≤ j ≤ N and N ≥ 3 and $DPE_i = DPE_j$ | DisjointDataProperties ( $DPE_1$ ... $DPE_i$ ... $DPE_N$) and 1 ≤ i ≤ N and N ≥ 2 |
| 5 | V | DisjointDataProperties( $DPE_1$ ... $DPE_N$ ) and 1 ≤ i ≤ N and N ≥ 2 | DisjointDataProperties( $DPE_i$ $DPE_j$ ) and i,j ∈ {1,N} and i ≠ j and N ≥ 2 |
| 6 | IV | DataPropertyDomain( DPE CE ) | SubClassOf( DataSomeValuesFrom (DPE rdfs:Literal ) CE ) |
| 7 | IV | DataPropertyRange( DPE DR ) | SubClassOf( owl:Thing DataAllValuesFrom( DPE DR ) ) |
| 8 | IV | FunctionalDataProperty( DPE ) | SubClassOf( owl:Thing DataMaxCardinality( 1 DPE ) ) |

*d) Assertion axioms*

OWL 2 *Assertion*[2] axioms are used to state facts about individuals. Following structural specification [2], the abstract class *Assertion* is specified by the following concrete classes: *SameIndividual*, *DifferentIndividuals*, *ClassAssertion*, *ObjectPropertyAssertion*, *NegativeObjectPropertyAssertion*, *DataPropertyAssertion*, *NegativeDataPropertyAssertion*. In Table 4, all transformations are proposed by us.

*Table 4:* The replaced and replacing assertion axioms.

| ID | Group | Replaced axiom | Replacing axiom(s) |
|---|---|---|---|
| 1 | II | SameIndividual( $a_1 \dots a_i \dots a_j \dots a_N$) and $1 \le i \le j \le N$ and $N \ge 3$ and $a_i = a_j$ | SameIndividual( $a_1 \dots a_i \dots a_N$) and $1 \le i \le N$ and $N \ge 2$ |
| 2 | V | SameIndividual( $a_1 \dots a_N$) and $1 \le i \le N$ and $N \ge 2$ | SameIndividual( $a_i\ a_j$ ) and $i,j \in \{1,N\}$ and $i \ne j$ and $N \ge 2$ |
| 3 | II | DifferentIndividuals( $a_1 \dots a_i \dots a_j \dots a_N$) and $1 \le i \le j \le N$ and $N \ge 3$ and $a_i = a_j$ | DifferentIndividuals( $a_1 \dots a_i \dots a_N$) and $1 \le i \le N$ and $N \ge 2$ |
| 4 | V | DifferentIndividuals( $a_1 \dots a_N$) and $1 \le i \le N$ and $N \ge 2$ | DifferentIndividuals( $a_i\ a_j$ ) and $i,j \in \{1,N\}$ and $i \ne j$ and $N \ge 2$ |

*e) Data ranges*

OWL 2 data ranges [2] are used in restrictions on data properties. The abstract class *DataRange* is specified by the following concrete classes: *DataComplementOf*, *DataUnionOf*, *DataOneOf*, *Datatype*, *DatatypeRestriction* and *DataIntersectionOf*. In Table 5, all transformations are our own proposals.

*Table 5:* The replaced and replacing data ranges.

| ID | Group | Replaced data range | Replacing data range(s) |
|---|---|---|---|
| 1 | III (3) | DataComplementOf (DataComplementOf( DR ) ) | DR |
| 2 | II | DataUnionOf( $DR_1 \dots DR_i \dots DR_j \dots DR_N$ ) and $1 \le i \le j \le N$ and $N \ge 3$ and $DR_j = DR_i$ | DataUnionOf( $DR_1 \dots DR_i \dots DR_N$ ) and $1 \le i \le N$ and $N \ge 2$ |
| 3 | III (1) | DataUnionOf (DataUnionOf( $DR_1 \dots DR_{Ai} \dots DR_{AN}$ ) ... $DR_{Bj} \dots DR_{BM}$ ) ) and $1 \le i \le N$ and $N \ge 2$ and $1 \le j \le M$ and $M \ge 2$ | DataUnionOf ($DR_1 \dots DR_{Ai} \dots DR_{AN} \dots DR_{Bj} \dots DR_{BM}$ ) ) and $1 \le i \le N$ and $N \ge 2$ and $1 \le j \le M$ and $M \ge 2$ |
| 4 | II | DataIntersectionOf ($DR_1 \dots DR_i \dots DR_j \dots DR_N$ ) and $1 \le i \le j \le N$ and $N \ge 3$ and $DR_j = DR_i$ | DataIntersectionOf( $DR_1 \dots DR_i \dots DR_N$ ) and $1 \le i \le N$ and $N \ge 2$ |
| 5 | III (1) | DataIntersectionOf (DataIntersectionOf( $DR_1 \dots DR_{Ai} \dots DR_{AN}$ ) ... $DR_{Bj} \dots DR_{BM}$ ) ) and $1 \le i \le N$ and $N \ge 2$ and $1 \le j \le M$ and $M \ge 2$ | DataIntersectionOf ($DR_1 \dots DR_{Ai} \dots DR_{AN} \dots DR_{Bj} \dots DR_{BM}$ ) ) and $1 \le i \le N$ and $N \ge 2$ and $1 \le j \le M$ and $M \ge 2$ |
| 6 | III (3) | DataIntersectionOf (DataComplementOf( $DR_1$ ) ... DataComplementOf( $DR_N$ ) ) and $1 \le i \le N$ and $N \ge 2$ | DataComplementOf (DataUnionOf( $DR_1 \dots DR_N$ ) ) and $1 \le i \le N$ and $N \ge 2$ |
| 7 | III (3) | DataUnionOf (DataComplementOf( $DR_1$ ) ... DataComplementOf( $DR_N$ ) ) and $1 \le i \le N$ and $N \ge 2$ | DataComplementOf (DataIntersectionOf( $DR_1 \dots DR_N$ ) ) and $1 \le i \le N$ and $N \ge 2$ |
| 8 | II | DataOneOf( $lt_1 \dots lt_i\ lt_j \dots lt_N$ ) and $1 \le i \le j \le N$ and $N \ge 1$ and $lt_j = lt_i$ | DataOneOf( $lt_1 \dots lt_i \dots lt_N$ ) and $1 \le i \le N$ and $N \ge 1$ |

*f) Class expressions*

OWL 2 class expressions are constructed of classes and properties. In structural specification [2] class expressions are represented by the *ClassExpression* abstract class. The abstract class *ClassExpression* is specified by the following concrete classes: *Class*, *ObjectIntersectionOf*, *ObjectUnionOf*, *ObjectComplementOf*, *ObjectOneOf*, *ObjectSomeValuesFrom*, *ObjectAllValuesFrom*, *ObjectHasValue*, *ObjectHasSelf*, *ObjectMinCardinality*, *ObjectMaxCardinality*, *ObjectExactCardinality*, *DataSomeValuesFrom*, *DataAllValuesFrom*, *DataHasValue*, *DataMinCardinality*, *DataMaxCardinality* and *DataExactCardinality*. In Table 6, the transformations of IDs: 9-14 and 19 are defined in [2], the other transformations are our proposal.

We exclude two general cases from further considerations - those of the existential and universal class expressions:

- *DataSomeValuesFrom( $DPE_1$ ... $DPE_N$ DR )*, where N ≥ 2 and
- *DataAllValuesFrom( $DPE_1$ ... $DPE_N$ DR )*, where N ≥ 2.

The reason is that in both class expressions the data range DR arity *MUST* be N (N≥ 2). However, the current version of OWL 2 specification [2] does not provide any constructor, which may be used to define data ranges of arity more than one (see Section 7 of [2]). If a future version of the specification provides such a constructor, one can consider removal of duplicates and further restructuration of the mentioned class expressions.

*Table 6:* The replaced and replacing class expressions.

| ID | Group | Replaced class expression | Replacing class expression(s) |
|---|---|---|---|
| 1 | III (3) | ObjectComplementOf<br>(ObjectComplementOf( CE ) ) | CE |
| 2 | II | ObjectUnionOf( $CE_1$ ... $CE_i$ ... $CE_j$ ... $CE_N$ )<br>and 1 ≤ i ≤ j ≤ N and N ≥ 3 and $CE_i$ = $CE_j$ | ObjectUnionOf( $CE_1$ ... $CE_i$ ... $CE_N$ )<br>and 1 ≤ i ≤ N and N ≥ 2 |
| 3 | III (1) | ObjectUnionOf<br>(ObjectUnionOf( $CE_1$ ... $CE_{Ai}$ ... $CE_{AN}$ )<br>... $CE_{Bj}$ ... $CE_{BM}$ ) )<br>and 1 ≤ i ≤ N and N ≥ 2<br>and 1 ≤ j ≤ M and M ≥ 2 | ObjectUnionOf<br>($CE_1$ ... $CE_{Ai}$ ... $CE_{AN}$ ... $CE_{Bj}$ ... $CE_{BM}$ ) )<br>and 1 ≤ i ≤ N and N ≥ 2<br>and 1 ≤ j ≤ M and M ≥ 2 |
| 4 | II | ObjectIntersectionOf<br>($CE_1$ ... $CE_i$... $CE_j$ ... $CE_N$ )<br>and 1 ≤ i ≤ j ≤ N and N ≥ 3 and $CE_i$ = $CE_j$ | ObjectIntersectionOf( $CE_1$ ... $CE_i$ ... $CE_N$ )<br>and 1 ≤ i ≤ N and N ≥ 2 |
| 5 | III (1) | ObjectIntersectionOf<br>(ObjectIntersectionOf )<br>($CE_1$ ... $CE_{Ai}$ ... $CE_{AN}$ )<br>... $CE_{Bj}$ ... $CE_{BM}$ ) )<br>and 1 ≤ i ≤ N and N ≥ 2<br>and 1 ≤ j ≤ M and M ≥ 2 | ObjectIntersectionOf<br>($CE_1$ ... $CE_{Ai}$ ... $CE_{AN}$ ... $CE_{Bj}$ ... $CE_{BM}$ ) )<br>and 1 ≤ i ≤ N and N ≥ 2<br>and 1 ≤ j ≤ M and M ≥ 2 |
| 6 | III (3) | ObjectIntersectionOf<br>(ObjectComplementOf( $CE_1$ )<br>... ObjectComplementOf( $CE_N$ ) )<br>and 1 ≤ i ≤ N and N ≥ 2 | ObjectComplementOf<br>(ObjectUnionOf( $CE_1$ ... $CE_N$ ) )<br>and 1 ≤ i ≤ N and N ≥ 2 |
| 7 | III (3) | ObjectUnionOf<br>(ObjectComplementOf( $CE_1$ )<br>... ObjectComplementOf( $CE_N$ ) )<br>and 1 ≤ i ≤ N and N ≥ 2 | ObjectComplementOf<br>(ObjectIntersectionOf( $CE_1$ ... $CE_N$ ) )<br>and 1 ≤ i ≤ N and N ≥ 2 |
| 8 | II | ObjectOneOf( $a_1$ ... $a_i$... $a_j$ ... $a_N$ )<br>and 1 ≤ i ≤ j ≤ N and N ≥ 1 and $a_i$ = $a_j$ | ObjectOneOf( $a_1$ ... $a_i$ ... $a_N$ )<br>and 1 ≤ i ≤ N and N ≥ 1 |
| 9 | IV | ObjectSomeValuesFrom( OPE CE ) | ObjectMinCardinality( 1 OPE CE ) |
| 10 | IV | ObjectAllValuesFrom( OPE CE ) | ObjectMaxCardinality<br>(0 OPE ObjectComplementOf( CE ) ) |
| 11 | IV | ObjectHasValue( OPE a ) | ObjectSomeValuesFrom<br>(OPE ObjectOneOf( a ) ) |
| 12 | IV | DataSomeValuesFrom( DPE DR ) | DataMinCardinality( 1 DPE DR ) |
| 13 | IV | DataAllValuesFrom( DPE DR ) | DataMaxCardinality<br>(0 DPE DataComplementOf( DR ) ) |
| 14 | IV | DataHasValue( DPE lt ) | DataSomeValuesFrom<br>(DPE DataOneOf( lt ) ) |
| 15 | III (2) | ObjectUnionOf<br>(ObjectMinCardinality( $n_1$ OPE CE )<br>( ObjectMinCardinality( $n_2$ OPE CE )<br>$CE_i$... $CE_N$ )<br>and 1 ≤ i ≤ N and N ≥ 3<br>and $n_1$≥ 0 and $n_2$≥ 0 and $n_1$ ≤ $n_2$ | ObjectUnionOf<br>(ObjectMinCardinality( $n_1$ OPE CE )<br>$CE_i$... $CE_N$ )<br>and 1 ≤ i ≤ N and N ≥ 2 and $n_1$≥ 0 |
| 16 | III (2) | ObjectIntersectionOf<br>(ObjectMinCardinality( $n_1$ OPE CE )<br>ObjectMinCardinality( $n_2$ OPE CE )<br>$CE_i$... $CE_N$)<br>and 1 ≤ i ≤ N and N ≥ 3<br>and $n_1$≥ 0 and $n_2$≥ 0 and $n_1$ ≤ $n_2$ | ObjectIntersectionOf<br>(ObjectMinCardinality( $n_2$ OPE CE )<br>$CE_i$... $CE_N$ )<br>and 1 ≤ i ≤ N and N ≥ 2 and $n_2$≥ 0 |
| 17 | III (2) | ObjectUnionOf<br>(ObjectMaxCardinality( $m_1$ OPE CE ) | ObjectUnionOf<br>(ObjectMaxCardinality( $m_2$ OPE CE ) |

| | | | |
|---|---|---|---|
| | | ObjectMaxCardinality( $m_2$ OPE CE )<br>$CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 3$<br>and $m_1 \geq 0$ and $m_2 \geq 0$ and $m_1 \leq m_2$ | $CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 2$ and $m_2 \geq 0$ |
| 18 | III (2) | ObjectIntersectionOf<br>( ObjectMaxCardinality( $m_1$ OPE CE )<br>ObjectMaxCardinality( $m_2$ OPE CE )<br>$CE_i$... $CE_N$)<br>and $1 \leq i \leq N$ and $N \geq 3$<br>and $m_1 \geq 0$ and $m_2 \geq 0$ and $m_1 \leq m_2$ | ObjectIntersectionOf<br>(ObjectMaxCardinality( $m_1$ OPE CE )<br>$CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 2$<br>and $m_1 \geq 0$ |
| 19 | IV | ObjectExactCardinality( n OPE CE )<br>and $n \geq 0$ | ObjectIntersectionOf<br>(ObjectMinCardinality( n OPE CE )<br>ObjectMaxCardinality( n OPE CE ) ) |
| 20 | III (2) | ObjectUnionOf<br>(DataMinCardinality( $n_1$ DPE DR )<br>DataMinCardinality( $n_2$ DPE DR )<br>$CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 3$ and $n_1 \leq n_2$<br>and $n_1 \geq 0$ and $n_2 \geq 0$ | ObjectUnionOf<br>(DataMinCardinality( $n_1$ DPE DR )<br>$CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 2$ and $n_1 \geq 0$ |
| 21 | III (2) | ObjectIntersectionOf<br>(DataMinCardinality( $n_1$ DPE DR )<br>DataMinCardinality( $n_2$ DPE DR )<br>$CE_i$... $CE_N$)<br>and $1 \leq i \leq N$ and $N \geq 3$<br>and $n_1 \geq 0$ and $n_2 \geq 0$ and $n_1 \leq n_2$ | ObjectIntersectionOf<br>(DataMinCardinality( $n_2$ DPE DR )<br>$CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 2$<br>and $n_2 \geq 0$ |
| 22 | III (2) | ObjectUnionOf<br>(DataMaxCardinality( $m_1$ DPE DR )<br>DataMaxCardinality( $m_2$ DPE DR )<br>$CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 3$<br>and $m_1 \geq 0$ and $m_2 \geq 0$ and $m_1 \leq m_2$ | ObjectUnionOf<br>(DataMaxCardinality( $m_2$ DPE DR )<br>$CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 2$ and $m_2 \geq 0$ |
| 23 | III (2) | ObjectIntersectionOf<br>(DataMaxCardinality( $m_1$ DPE DR )<br>DataMaxCardinality( $m_2$ DPE DR )<br>$CE_i$... $CE_N$)<br>and $1 \leq i \leq N$ and $N \geq 3$<br>and $m_1 \geq 0$ and $m_2 \geq 0$ and $m_1 \leq m_2$ | ObjectIntersectionOf<br>(DataMaxCardinality( $m_1$ DPE DR )<br>$CE_i$... $CE_N$ )<br>and $1 \leq i \leq N$ and $N \geq 2$ and $m_1 \geq 0$ |
| 24 | IV | DataExactCardinality( n DPE DR )<br>and $n \geq 0$ | ObjectIntersectionOf<br>(DataMinCardinality( n DPE DR )<br>(DataMaxCardinality( n DPE DR ) ) |

*g) Object property expressions*

The following OWL 2 structural specification [2] object property expressions are represented by *ObjectPropertyExpression* abstract class. The abstract class *ObjectPropertyExpression* is specified by the following concrete classes: *ObjectProperty* and *InverseObjectProperty*. In Table 7, the transformation is our proposal.

*Table 7:* The replaced and replacing object property expressions.

| ID | Group | Replaced object property expression | Replacing object property expression |
|---|---|---|---|
| 1 | III (3) | ObjectInverse of( ObjectInverse Of ( OP ) ) | OP |

## V. Ontology Normalization Algorithm

The following is an outline of the algorithm which transforms the syntactically correct and consistent OWL 2 DL ontology selected by the user – denoted by $OWL_{ONT}$ – into the normalized ontology. The $OWL_{ONT}$' and $OWL_{ONT}$'' are intermediate ontologies required to process the input ontology into the output ontology. In the beginning, both $OWL_{ONT}$' and $OWL_{ONT}$'' are empty. On completion of the algorithm, the $OWL_{ONT}$'' represents the normalized ontology.

*Algorithm:* Outline of the ontology normalization algorithm

*Input:* Syntactically correct and consistent OWL 2 DL ontology

*Output:* Normalized OWL 2 DL ontology

*BEGIN*

1. Take the first axiom from $OWL_{ONT}$.
2. Take the first entity from the selected axiom.
3. If the entity is declared, add the declaration axiom to $OWL_{ONT}$'.If the entity is not declared, extract the declaration axiom for the entity based on its usage and add the new declaration axiom to $OWL_{ONT}$'.
4. Take the next entity from the selected axiom.
5. Repeat steps 3-4 until no more entities in the selected axiom are available.
6. Apply to the selected axiom allapplicable replacement rules defined in Tables 5-7, receiving a modified axiom.
7. Add the modified axiom to $OWL_{ONT}$'.
8. Take the next axiom from $OWL_{ONT}$.
9. Repeat steps 2-8 until no more axioms in $OWL_{ONT}$ are available.
10. Take the first axiom from $OWL_{ONT}$'.
11. Apply to the axiom allapplicable replacement rules defined in Tables 1-4.
12. If transformations result in only one axiom, add the axiom to $OWL_{ONT}$". Otherwise, if as a result of transformations the axiom splits into two or more axioms, repeat step 11 for each split axiom independently.
13. Take the next axiom from $OWL_{ONT}$'.
14. Repeat steps 11-13 until no more axioms in $OWL_{ONT}$' are available.
15. Eliminate any of the duplicated axioms from $OWL_{ONT}$" ontology.
16. Return the $OWL_{ONT}$" as a normalized ontology.

*END*

*Comments to the algorithm:*

1. OWL 2 ontologies are built of axioms which may contain some expressions. Data ranges are contained in two axioms: *DatatypeDefinition* and *DataPropertyRange*, as well as in some expressions, e.g.,*DataAllValuesFrom*, *DataMinCardinality*, etc. Therefore, to perform fewer iterations of the normalization algorithm, first, we conduct all the transformations of the data ranges in axioms and expressions, as well as the expressions in axioms, and later on of the axioms themselves.
2. If the input ontology does not contain any duplicated axioms, the resulting ontology will contain at least the same number of axioms as the input ontology.
3. The order of the conducted transformations is not important because the resulting ontology will always be semantically equivalent. However, depending on the selected order, the resulting ontology may have a different textual form. The possible textual differences in the output ontology include: (1) the order of axioms and (2) the order of expressions in axioms (only if the order of expressions in the selected axiom is not important).
4. The resulting ontology may contain fewer kinds of axioms and expressions. In particular, the ontology will not contain the below-mentioned list of axioms and expressions because they are refactored and reduced in accordance with the presented transformations:
   - *Class axioms: EquivalentClasses*, *DisjointClasses*, *DisjointUnion*,
   - *Object property axioms: EquivalentObjectProperties*, *InverseObjectProperties*, *ObjectPropertyDomain*, *ObjectPropertyRange, InverseFunctionalObject Property*, *FunctionalObjectProperty*, *ReflexiveObject Property*, *IrreflexiveObjectProperty*, *SymmetricObject Property*,*TransitiveObjectProperty*,
   - *Data property axioms: EquivalentDataProperties*, *DataPropertyDomain*, *DataPropertyRange*, *Functional DataProperty*,
   - *Class expressions: ObjectSomeValuesFrom*, *Object AllValuesFrom*, *ObjectHasValue*, *ObjectExact Cardinality*, *DataSomeValuesFrom*, *DataAllValues From*, *DataHasValue*, *DataExactCardinality*.
5. The method of normalization and the defined transformations are unidirectional, which means that it is not possible to retrieve the original ontology from the normalized ontology.

## VI. Example of Single Normalization

The example presents transformations conducted with the use of the normalization algorithm. The following is an input ontology, which contains one axiom:

EquivalentClasses(:FourLeafClover: FourLeafClover ObjectIntersectionOf(
ObjectMinCardinality(3: hasLeaf:Leaf ) ObjectMaxCardinality( 7 :hasLeaf :Leaf )
ObjectExactCardinality( 4 :hasLeaf :Leaf ) ) )

Steps 1-5 of the algorithm extract declarations of entities:

Declaration( Class ( :FourLeafClover ) ) (1)
Declaration( Class ( :Leaf ) ) (2)
Declaration( ObjectProperty (:hasLeaf ) ) (3)

Steps 6-9 of the algorithm result in the following transformations:

*Rule 19 from Table 6* applied on the given axiom (4)
EquivalentClasses(:FourLeafClover :FourLeafClover ObjectIntersectionOf(
ObjectMinCardinality( 3 :hasLeaf :Leaf )
ObjectMaxCardinality( 7 :hasLeaf :Leaf )
ObjectIntersectionOf( ObjectMinCardinality( 4 :hasLeaf :Leaf )
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) )

*Rule 5 from Table 6* applied on (4) (5)
EquivalentClasses( :FourLeafClover :FourLeafClover ObjectIntersectionOf(
ObjectMinCardinality( 3 :hasLeaf :Leaf )
ObjectMaxCardinality( 7 :hasLeaf :Leaf )
ObjectMinCardinality( 4 :hasLeaf :Leaf )
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) )

*Rule 20 from Table 6* applied on (5) (6)
EquivalentClasses( :FourLeafClover :FourLeafClover
ObjectIntersectionOf( ObjectMaxCardinality( 7 :hasLeaf :Leaf )
ObjectMinCardinality( 4 :hasLeaf :Leaf )
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) )

*Rule 23 from Table 6* applied on (6) (7)
EquivalentClasses( :FourLeafClover :FourLeafClover
ObjectIntersectionOf( ObjectMinCardinality( 4 :hasLeaf :Leaf )
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) )

Steps 10-15 of the algorithm result in the following transformations:

*Rule 1 from Table 1* applied on (7) (8)
EquivalentClasses( :FourLeafClover ObjectIntersectionOf(
ObjectMinCardinality( 4 :hasLeaf :Leaf )
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) )

*Rule 2 from Table 1* applied on (8) (9)
SubClassOf( :FourLeafClover ObjectIntersectionOf(
ObjectMinCardinality( 4 :hasLeaf :Leaf )
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) )
SubClassOf( ObjectIntersectionOf( ObjectMinCardinality( 4 :hasLeaf :Leaf )
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) :FourLeafClover )

Steps 16-17 of the algorithm return the normalized ontology:

Declaration( Class ( :FourLeafClover ) ) (1)
Declaration( Class ( :Leaf ) ) (2)
Declaration( ObjectProperty ( :hasLeaf ) ) (3)
SubClassOf( :FourLeafClover ObjectIntersectionOf( (9A)
ObjectMinCardinality( 4 :hasLeaf :Leaf )
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) )
SubClassOf( ObjectIntersectionOf( ObjectMinCardinality( 4 :hasLeaf :Leaf ) (9B)
ObjectMaxCardinality( 4 :hasLeaf :Leaf ) ) :FourLeafClover )

## VII. Proofs of the Correctness of the OWL 2 Construct Replacements

This section aims at presenting proofs of correctness of the OWL 2 construct replacements presented in tables in Section 4. The replacing language constructs (right column in tables) are semantically equivalent to the replaced language constructs (left column in tables).

The proofs are based on direct model-theoretic semantics [13]for OWL 2, which is compatible with the description logic *SROIQ*. The following convention is used:

1. $V_C$ is a set of classes containing at least the owl: Thing and owl: Nothing classes.
2. $V_{OP}$ is a set of object properties containing at least the object properties owl: topObjectProperty and owl: bottomObjectProperty.
3. $\Delta_I$ is a nonempty set called the object domain.
4. $(\ )^C$ is the class interpretation function that assigns to each class $C \in V_C$ a subset $(C)^C \subseteq \Delta_I$ such that $(\text{owl: Thing})^C = \Delta_I$ and $(\text{owl: Nothing})^C = \emptyset$
5. $(\ )^{OP}$ is the object property interpretation function that assigns to each object property $OP \in V_{OP}$ a subset $(OP)^{OP} \subseteq \Delta_I \times \Delta_I$ such that $(\text{owl:topObjectProperty})^{OP} = \Delta_I \times \Delta_I$ and $(\text{owl: bottomObjectProperty})^{OP} = \emptyset$
6. $\alpha = \beta$ means semantic equivalence of $\alpha$ and $\beta$ sets.
7. $\alpha \vDash B$ means that $\alpha$ formula is the semantic consequence of $B$ set of formulas.

Proving equivalence comes down to the use of the interpretation definition and the rules of set theory. We selected two replacement rules for the proofs; all other ones could be proved analogically.

*Proof 1* for construct replacements from Table 1 ID 6:
We have to prove that the interpretation of

DisjointClasses( $CE_1$ $CE_2$ )

is equivalent to the interpretation of

SubClassOf( $CE_1$ ObjectComplementOf( $CE_2$ ) )

The interpretation of

DisjointClasses( $CE_1$ $CE_2$ )

is (1) [13]:

$$(CE_1)^C \cap (CE_2)^C = \emptyset \quad (1)$$

The interpretation of

ObjectComplementOf( $CE_2$ )

is (2) [13]:

$$\Delta_I \setminus (CE_2)^C \quad (2)$$

The interpretation of

SubClassOf( $CE_1$ $CE_3$ )

is (3) [13]:

$$(CE_1)^C \subseteq (CE_3)^C \quad (3)$$

Based on (2) and (3) the interpretation of

SubClassOf( $CE_1$ ObjectComplementOf( $CE_2$ ) )

is (4):

$$(CE_1)^C \subseteq \Delta_I \setminus (CE_2)^C \quad (4)$$

We have to show that (4) is correct.

If we assume that (4) is false, it means that (5) is true:

$$(CE_1)^C \not\subseteq \Delta_I \setminus (CE_2)^C \quad (5)$$

It means that there exist:

$$x \in (CE_1)^C \wedge x \notin \Delta_I \setminus (CE_2)^C \Leftrightarrow$$
$$x \notin \Delta_I \setminus (CE_2)^C \Rightarrow x \in (CE_2)^C$$

Then:

$$x \in (CE_1)^C \wedge x \in (CE_2)^C \Leftrightarrow$$
$$x \in (CE_1)^C \cap (CE_2)^C$$

It means that:

$$(CE_1)^C \cap (CE_2)^C \neq \emptyset$$

We have received contradiction, which had to be proved.

*Proof 2* for construct replacements from Table 6 ID 7:
We have to prove that the interpretation of

ObjectUnionOf(
ObjectComplementOf( $CE_1$ )

...
ObjectComplementOf( $CE_N$ ) )

where $1 \leq i \leq N$ and $N \geq 2$ is equivalent to the interpretation of

ObjectComplementOf( ObjectIntersectionOf( $CE_1$ ... $CE_N$ ) )

where $1 \leq i \leq N$ and $N \geq 2$.

The interpretation of

ObjectUnionOf( $CE_1$ ... $CE_N$ )

is (14) [13]:

$$(CE_1)^C \cup \ldots \cup (CE_n)^C \tag{14}$$

The interpretation of

ObjectIntersectionOf( $CE_1$ ... $CE_n$ )

is (15) [13]:

$$(CE_1)^C \cap \ldots \cap (CE_n)^C \tag{15}$$

Based on De Morgan's law for sets, (2) and (14) the interpretation of

ObjectUnionOf(
ObjectComplementOf( $CE_1$ )
...
ObjectComplementOf( $CE_N$ ) )

is (16):

$$(\triangle_I \setminus (CE_1)^C) \cup \ldots \cup (\triangle_I \setminus (CE_N)^C) \tag{16}$$

(17) is a result of application of (16) to (17):

$$\triangle_I \setminus ((CE_1)^C \cap \ldots \cap (CE_N)^C) \tag{17}$$

Based on (2) and (15) interpretation of

ObjectComplementOf( ObjectIntersectionOf( $CE_1$ ... $CE_N$ ) )

is (18):

$$\triangle_I \setminus ((CE_1)^C \cap \ldots \cap (CE_N)^C) \tag{18}$$

The equations (17) and (18) are equal, which had to be proved.

## VIII. Conclusions

The paper introduces the concept of ontology normalization as a process of transforming the input OWL 2 ontology into the ontology in its refactored form. The process is defined through a group of OWL 2 construct replacements. Because all individual replacing constructs preserve the semantics of the replaced constructs, the resulting ontology does not change the semantics of the original ontology.

Thanks to the presented approach, users obtain the possibility to automate the processing of ontologies because the normalized ontologies have the structure of axioms unified. However, the normalized ontology has reduced readability from the point of view of human readers, which is caused especially by the transformations from the group IV, which remove the syntactic sugar from the ontology.

The presented normalization algorithm is implemented in a prototype tool [8] which additionally allows for comparing two ontologies with the agreed vocabulary. More specifically, the tool states whether or not two ontologies are compliant or contradictory by the method outlined in [6].

## References Références Referencias


1. OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation 11 December 2012. https://www.w3.org/TR/owl2-overview/. 2012.
2. 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/. 2012.
3. I Horrocks, O. Kutz, and U. Sattler, "The Even More Irresistible SROIQ," Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006). AAAI Press, pp. 57–67, 2006.
4. OWL 2 Web Ontology Language New Features and Rationale (Second Edition) W3C Recommendation 11 December 2012, https://www.w3.org/TR/owl2-new-features/. 2012.
5. OWL 2 Web Ontology Language Profiles (Second Edition). W3C Recommendation 11 December 2012. https://www.w3.org/TR/owl2-profiles/. 2012.
6. M. Sadowska and Z. Huzar, "Semantic Validation of UML Class Diagrams with the Use of Domain Ontologies Expressed in OWL 2," Software Engineering: Challenges and Solutions. Springer International Publishing, pp. 47–59, 2017.
7. OMG, Unified Modeling Language,Version 2.5, Doc. No: ptc/2013-09-05, http://www.omg.org/spec/UML/2.5. 2015.
8. M. Sadowska, "A Prototype Tool for Semantic Validation of UML Class Diagrams with the Use of Domain Ontologies Expressed in OWL 2," In Towards a Synergistic Combination of Research

and Practice in Software Engineering. Springer, Cham, pp. 49–62, 2018.
9. V. Denny and Y. Sure, "How to design better ontology metrics," The Semantic Web: Research and Applications, pp. 311–325, 2007.
10. A. L. Rector, "Normalisation of ontology implementations: Towards modularity, re-use, and maintainability," Proceedings Workshop on Ontologies for Multiagent Systems (OMAS) in conjunction with European Knowledge Acquisition Workshop, pp. 1–16, 2002.
11. A. L. Rector, "Modularisation of domain ontologies implemented in description logics and related formalisms including OWL," Proceedings of the 2nd international conference on Knowledge capture. ACM, pp. 121–128, 2003.
12. Meta Object Facility (MOF) Core Specification, version 2.0. Object Management Group, OMG, http://www.omg.org/spec/MOF/2.0/PDF/. 2006.
13. OWL 2 Web Ontology Language Direct Semantics (Second Edition) W3C Recommendation 11 December 2012, https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/. 2012.