Computer Science and Information Systems 22(1):105–131 https://doi.org/10.2298/CSIS240227002H

Formal Transformation of OWL Ontology to a FOKI Generic Meta-model

Bogumila Hnatkowska, Adrianna Kozierkiewicz, and Marcin Pietranik

Wroclaw University of Science and Technology,
Wybrzeże Wyspiańskiego 27, 50-370, Wrocław, Poland
{bogumila.hnatkowska, adrianna.kozierkiewicz, marcin.pietranik}@pwr.edu.pl

Abstract. Ontology integration is merging a set of ontologies to provide a single, unified ontology, which contains all of the knowledge from input ontologies. Most solutions described in the literature are based on the OWL format and incorporate its strengths and weaknesses. In our previous research, we developed the ontology integration framework FOKI, which does not use the OWL. Collected experimental data using prepared ontologies proved its usefulness. However, the lack of OWL support makes it challenging to use the FOKI framework in practical applications. This paper presents a meta-model and a set of transformation rules for bi-directional transformation between ontologies expressed in our framework and the OWL standard. The meta-model serves as a bridge in the transformation process. Transformation rules are built by referencing an abstract syntax element of OWL2 and an appropriate mathematical formalism from FOKI. Their correctness was verified on widely available ontologies expressed in OWL provided, e.g., by the Ontology Alignment Evaluation Initiative.

Keywords: FOKI, ontology integration, OWL, transformation

1. Introduction

Ontology integration, a well-explored area within the field of knowledge management, addresses the challenge of merging multiple source ontologies into a unified target ontology, encapsulating the collective knowledge of the input sources. This process involves selecting elements from source ontologies representing the same aspects of a modeled universe of discourse and integrating them into a cohesive entity while preserving unique, non-conflicting elements. Formally, it can be defined as: *for the given n ontologies $O_1, O_2, ..., O_n$ one should determine an ontology O^*, which is the best representation of the given input ontologies.*

The topic is widely discussed in literature. In [20], authors examine the challenge of semantic heterogeneity in ontologies on the Semantic Web, emphasizing the importance of ontology alignment and merging for achieving interoperability. It categorizes existing merging approaches into three types—single-strategy, multiple-strategy, and those utilizing external semantic resources—and introduces a novel framework that leverages multiple knowledge bases to address semantic discrepancies and enhance ontology merging.

For example, in [6], authors address the need for effective methodologies to integrate and reuse ontologies, which are vital for standardizing information and fostering knowledge-based digital ecosystems. The paper introduces an approach that employs heterogeneous matching techniques to automate the process of building ontologies, thereby simplifying the creation of digital ecosystems. The method has been successfully applied to the food production domain, demonstrating its practical utility.

The article [24] discusses the integration of biomedical ontologies using Semantic Web standards, specifically addressing the challenges posed by structural differences that can lead to inconsistencies when merging. It introduces a framework that uses a seed ontology for iterative enrichment with new sources, employing a novel fine-grained approach for mapping repair and alignment conservativity supported by both theoretical formalization and practical algorithms. The framework has been applied to integrate multiple medical ontologies, enhancing real-world healthcare services provided by Babylon Health, and shows promising results when compared with leading ontology integration systems.

The academic literature is replete with various approaches to address this topic, each offering unique insights and methodologies. A comprehensive review of these diverse strategies and an exploration of ongoing research avenues that have yet to be fully explored is described in detail in ([22]). This work not only aggregates the different available solutions but also critically examines their effectiveness and potential for further development.

While the concept of ontology integration is straightforward conceptually, it presents significant semantic and computational complexities. Most existing solutions are restricted to a specific ontology representation format, typically OWL (Web Ontology Language), particularly its second version, OWL2. Despite OWL's robustness, it imposes limitations on the expressiveness of concept attributes. For instance, once an attribute is associated with a concept, it cannot be reused with the same name in another context within the ontology, leading to rigid and unnatural naming conventions to circumvent this limitation.

Our research has identified that attributes often acquire varied meanings across different concepts. For example, the attribute ”number” might represent different data in the ”Book” and ”Address” contexts. This observation inspired us to develop a novel ontology expression and management approach based on a solid formal foundation that ensures flexible domain modeling and determinism. We proved in [23] that an approach to ontology alignment (and later into ontology integration) based on analyzing attributes semantics gives good results.

This research led to the development of the Framework for Ontological Knowledge Integration (FOKI), which offers a formally grounded, theoretically robust tool for ontology management. The FOKI framework allows for a detailed examination of ontological elements and their interrelationships, providing a comprehensive perspective on ontology structure. Moreover, our approach provides a broad perspective on ontologies, allowing a deep analysis of their internal elements and their relationships.

The developed framework has many advantages, such as limiting human participation in the integration process to a minimum, resolving issues related to a diversity of semantic relations, and enabling proper identification of mappings between the ontologies. Additionally, the FOKI framework can express changes that appear while maintained ontologies evolve. Informally speaking – a FOKI model can answer the question ”What changed when”. Several applications showed the usefulness of our approach in both the tasks of ontology integration and ontology alignment ([18], [19], [17]).

Despite its advantages, the primary limitation of FOKI is its incompatibility with OWL, particularly for practical applications that rely on standard ontology formats like those provided by the Ontology Alignment Evaluation Initiative (OAEI) [1] – a state-of-the-art dataset used to test the usefulness of many ontology-related applications. Therefore, we need to transform the FOKI model into the OWL standard for experimental verification and comparison of our approach with other solutions.

This paper, therefore, focuses on establishing a set of transformation rules that facilitate the conversion between OWL (OWL2 specifically) and the FOKI formalism, aiming to bridge the gap between these two frameworks and enhance the practical deployment of FOKI. Every rule must be built by referencing an abstract syntax element of OWL2 and an appropriate mathematical formalism of the FOKI framework (obviously if such exists and is applicable).

Preparing the aforementioned transformations presents several challenges. The first significant challenge arises from the assumption in FOKI that attributes and relations are annotated with logical sentences, explicitly defining their intended semantics. This requirement substantially increases the complexity of maintaining the FOKI ontology. Each logical sentence must be meticulously crafted and maintained to preserve the intended semantics, which can be time-consuming and error-prone. The explicit annotation of semantics also demands higher expertise from ontology developers, making developing and maintaining FOKI-based ontologies more demanding compared to other frameworks.

Secondly, the OWL standard operates under the premise of an open-world domain, where it is assumed that the absence of information does not imply falsehood. In contrast, FOKI assumes a closed-world domain, where what is unknown as true is considered false. This fundamental difference in assumptions can lead to an OWL ontology built using FOKI formalism being overly restrictive. For example, OWL provides mechanisms to express that two concepts are equivalent, implying that their sets of instances cannot be disjointed. In FOKI, it is straightforward to check whether two sets of instances are disjoint; however, this does not necessarily mean that the ontology developer intended these concepts to be equivalent. This mismatch in assumptions can result in unintended interpretations and constraints within the OWL ontology, complicating its application and potentially leading to incorrect inferences.

Eventually, we introduce a meta-model that encapsulates the essential concepts of FOKI. This meta-model is built on a dual foundation: it functions both as a simplified version of the OWL2 meta-model and as an extension that includes elements unique to FOKI. This dual nature facilitates bi-directional translation between OWL2 and FOKI, allowing for the transformation of ontologies in both directions with minimal information loss.

Our goal is to develop procedures that enable these transformations while preserving the integrity and semantics of the original ontologies. This involves addressing the significant semantic differences between OWL and FOKI, such as the open-world assumption in OWL versus the closed-world assumption in FOKI, as well as the unique name assumption in OWL and its reproduction in FOKI. By carefully considering these differences, we aim to create a robust and reliable method for ontology translation that maintains the fidelity of the original data.

The developed meta-model is a versatile solution that accommodates varying semantic definitions and relationships inherent in different ontologies, making it universally applicable. Its design allows it to adapt to and integrate diverse ontological structures and

[1] http://oaei.ontologymatching.org

semantics, ensuring that the FOKI framework is not limited to any specific domain or ontology structure. This flexibility supports a broad range of ontology types and integration scenarios, enhancing its utility across various applications.

By accommodating different semantic definitions, the meta-model ensures that the FOKI framework can handle the complexities of integrating multiple ontologies with differing structures and semantics. This universality is particularly beneficial for projects requiring data synthesis from disparate sources, enabling seamless interoperability and more comprehensive data analysis. Consequently, the FOKI framework becomes a powerful tool for a diverse range of applications, from academic research to industry-specific solutions, ensuring robust and efficient ontology management and integration.

The paper is organized as follows. Section 2 contains an overview of similar research found in the literature. Section 3 provides base mathematical definitions used throughout the paper. A motivating example is presented in Section 4 The core contribution is given in Section 5, and the accepted evaluation procedure with its results can be found in Section 6. Section 7 summarizes the paper and overviews our upcoming research plans.

2. Related Works

In real situations, there is a need to transform one model or standard into another. This paper is devoted to ontology transformation, which can be defined as changing one ontology format into another with the same semantics. In other words, we will define rules that allow us to run transformation between an OWL ontology and FOKI.

There exists different ontology representation approaches like Suo-Kif [21], OWL (expressed by many concrete syntaxes, e.g., functional, Manchester, XML, RDF Turtle, RDF/XML), database schema, different mathematical representations or UML. In many practical solutions ontologies, especially expressed in OWL2, are subjects of transformations to/from different notations [27], UML [8], [9], [27], [28],[7], description logic [5], programming languages [4], [12], database (schema) [1], [2], [14] or business vocabularies and rules [15], [16].

Many approaches exist to the definition of transformation rules, depending on their formality level and paradigms used. It is possible to represent transformation rules in natural language [1], [4], [8], [9], [12], [25]. The alternative approach is a semi-formal method that can be based, e.g., on structured tables [14], or patterns with placeholders [16]. The most precise method is based on a mathematical apparatus [2], [27]. It reduces the risk of misinterpretation but could be hard to understand. So the formal representations are typically extended with simple explanations.

The second division is based on how the transformation rules are written – one can use either a declarative ([28]) or operational language ([8]). Sometimes a hybrid method is used, e.g. [15]. When the ordering of transformation rules matters, their ordering is also provided [27].

We can also differ transformations based on the direction (one or two ways) and the loss of information (lossy transformation vs. lossless). Authors of [1] developed the OWLMap tool for the automatic and lossless ontology transformation into a relational database. Similarly, [25] proposed a transformation system providing a lossless roundtrip mapping for OBO and OWL ontologies. Rule-based transformations presented by [3] and [29] are lossy as some OWL axioms are not considered, e.g., property restrictions.

Similarly, in [8], only some features of UML class diagrams have been preserved in the transformation to OWL. Xu and et.all [27] manage to propose a semantics-preserving approach for extracting OWL ontologies from existing UML class diagrams, however, it cannot support several new syntaxes (including the Functional-Style syntax and the Manchester syntax) and richer datatypes of OWL2 language. Authors of [26] proposed an algorithm for transforming OWL ontology into a relational database that works without a loss of information; however, authors are aware of limitations to represent more advanced OWL features.

The created FOKI meta-model and transformation procedures allow us to transform ontologies (OWL2 – FOKI) in both directions almost losslessly, which is a major advantage. This issue is discussed in more detail in Section 5. The transformation procedures are defined in an operational manner.

3. Basic Notions

Framework for Ontological Knowledge Integration (FOKI) is built on mathematical foundations [23]. We assume that a real world is defined as a triple (A, V, P) where: A is a finite set of attributes that can be used to describe objects, V is a set of their valuations (domains) such that $V = \bigcup_{(a \in A)} V_a$, where V_a is a domain of a particular attribute a and P is a set of predicates that can be used to define integrity constraints. The following sextuple defines an ontology in FOKI as a (A, V, P)-based ontology:

$$O = (C, H, R^C, I, R^I, Z) \tag{1}$$

where: C is a set of concepts; H is a concepts' hierarchy; R^C is a set of relations between concepts, $R^C = \{r_1^C, r_2^C, ..., r_n^C\}$, $n \in N$, such that $r_i^C \in R^C (i \in [1, n])$ is a subset of $C \times C$; I is a set of instance identifiers; $R^I = \{r_1^I, r_2^I, ..., r_n^I\}$ is a set of relations between concepts' instances, Z is a set of first-order logic sentences, built over predicates from P. Each concept $c \in C$ is defined as:

$$c = (id^c, A^c, V^c, I^c) \tag{2}$$

where: id^c is an identifier of the concept c, A^c is a set of its attributes, such that $A^c \subseteq A$, V^c is a set of attributes domains (formally: $V^c = \bigcup_{(a \in A^c)} V_a$), I^c is a set of instances of the concept c. We write $a \in c$ to denote that an attribute a belongs to concept c set of attributes A^c. It is worthy of notice that attributes from set A have no semantics. They can be only interpreted if they are part of the chosen concept [10].

By D_A we denote a set containing of words that can be used to define the sublanguage L_S^A of the propositional calculus composed of D_A elements and logical operators of conjunction, disjunction, and negation. By their own, elements of D_A. can be interpreted as atomic descriptions of attributes. Formally, we can define a function S_A that assigns a logical sentence from L_S^A to attributes A^c within a specific concept c. The function S_A has a following signature $S_A : A \times C \to L_S^A$ and allows us to formally define relation: *equivalency* (denoted as $\equiv$), *generalization* (denoted as $\leftarrow$) and *contradiction* (denoted as $\sim$) between attributes:

- Two attributes $a \in A^{c_1}, b \in A^{c_2}$ are *semantically equivalent*, denote by $a \equiv b$, if the formula $S_A(a, c_1) \Leftrightarrow S_A(b, c_2)$ is a tautology for any two $c_1 \in C_1$ and $c_2 \in C_2$.

- The attribute $a \in A^{c_1}$ in concept $c_1 \in C_1$ is *more general* than the attribute $b \in A^{c_2}$ in concept $c_2 \in C_2$ (denoted by $a \leftarrow b$) if the formula $S_A(b, c_2) \Rightarrow S_A(a, c_1)$ is a tautology for any two $c_1 \in C_1$ and $c_2 \in C_2$.
- Two attributes $a \in A^{c_1}, b \in A^{c_2}$ are in *semantical contradiction*, denoted by $a \sim b$, if the formula $\neg(S_A(a, c_1) \wedge S_A(b, c_2))$ is a tautology for any two $c_1 \in C_1$ and $c_2 \in C_2$.

Instances are closely related to the concepts because they are their physical materialization. Thus, we can write $i \in c$, which can be read that the instance i belongs to the concept c. Formally, an instance i which belongs to the set I_C is defined as a pair $i = (id^i, v_c^i)$ where:

- id^i is an instance identifier,
- v_c^i is a function with a signature $v_c^i : A^c \to V^c$

Instances can belong to many different concepts, therefore, a set of instances' identifiers from Equation 1 is defined as $I = \bigcup_{c \in C} \{id^i | (id^i, v_c^i) \in I^c\}$. We also define an auxiliary function $Ins(c) = \{id^i | (id^i, v_c^i) \in I^c\}$ which for a given concept *c* returns a set containing identifiers of instances assigned to it.

By analogy to D_A we define a set D_R and a sub-language L_S^R to give relations semantics. Formally, we can define a function $S_R : R^C \to L_S^R$ that assigns a logical sentence from L_S^R to a relation from the set R^C. As a consequence, we can define formal criteria for relationships between relations, analogical to criteria for relationships between attributes. However, within L_S^R we distinguish elements: *is_symmetric*, *is_transitive* and *is_reflexive* that for some selected relation *r* can be used to describe their properties:

- $\big(S_R(r_1^C) \implies is\_symmetric\big) \Rightarrow (\forall (c_1, c_2) \in r_1^C : c_1 = c_2) \wedge \forall (i_1, i_2) \in r_1^I \; \exists (i_2, i_1) \in r_1^I$, where $c_1, c_2 \in C, r_1^C \in R^C, r_1^I \in R^I, i_1 \in c_1 \vee c_2, i_2 \in c_1 \vee c_2)$
- $\big(S_R(r_1^C) \implies is\_transitive\big) \Rightarrow (\forall (c_1, c_2) \in r_1^C : c_1 = c_2) \wedge \forall (i_1, i_2), (i_2, i_3) \in r_1^I$: $(i_1, i_3) \in r_1^I)$ where $c_1, c_2 \in C, r_1^C \in R^C, r_1^I \in R^I, i_1 \in c_1 \vee c_2, i_2 \in c_1 \vee c_2)$
- $\big(S_R(r_1^C) \implies is\_reflexive\big) \Rightarrow (\forall (c_1, c_2) \in r_1^C : \exists (c_1, c_1) \in r_1^C \wedge \exists (c_2, c_2) \in r_1^C) \wedge \forall (i_1, i_2) \in r_1^I : \exists (i_1, i_1) \in r_1^I \wedge \exists (i_2, i_2) \in r_1^I)$

In the above conditions, we utilize the $\implies$ symbol to denote the implication that occurs within logical sentences built from elements from L_S^R. The symbol $\Rightarrow$ denotes the implication in the context of the ontology definition and its elements. To simplify the notation, we will use predicates (e.g., $is\_asymmetric\,(r)$)) to denote properties defined above.

The hierarchy of concepts *H* is simply defined as a subset of a cartesian product of a set of concepts $H \subset C \times C$. A pair of concepts (c_1,c_2) such that $c_1 = (id^{c_1}, A^{c_1}, V^{c_1}, I^{c_1})$ and $c_2 = (id^{c_2}, A^{c_2}, V^{c_2}, I^{c_2})$ may be included in the hierarchy (to represents the fact that c_1 is an ancestor of c_2) only if all of the following postulates are met:

1. $|A^{c_2}| \geq |A^{c_1}|$
2. $\forall a \in A^{c_1} \exists a' \in A^{c_2} : (a \equiv a') \vee (a \leftarrow a')$
3. $Ins(c_2) \subseteq Ins(c_1)$

The set of relations R^C allows us to describe which concept instances can be connected. The actual materialization of these connections is described by the set of instance relations marked as R^I which must satisfy the following formal criteria:

1. $r_j^I \subseteq \bigcup_{(c_1,c_2)\in r_j^C} (Ins(c_1) \times Ins(c_2))$
2. $(i_1, i_2) \in r_j^I \implies \exists (c_1, c_2) \in r_j^C : (c_1 \in Ins^{-1}(i_1)) \wedge (c_2 \in Ins^{-1}(i_2))$ which describes that two instances may be connected by some relation only if there is a relation connecting concepts they belong to
3. $(i_1, i_2) \in r_j^I \implies \neg \exists r_k^I \in R^I : ((i_1, i_2) \in r_k^I) \wedge (r_j^C \sim r_k^C)$ which concerns a situation in which two instances cannot be connected by two contradicting relations
4. $(i_1, i_2) \in r_j^I \wedge \exists r_k^I \in R^I : r_k^C \leftarrow r_j^C \implies (i_1, i_2) \in r_k^I$ which denotes that if two instances are in a relation and there exists a more general relation, then they are also connected by it.

4. Motivating example

As was mentioned before, FOKI lacks a formal syntax but delivers – at the meta-model level – the necessary means for ontology integration. The semantics of attributes and relations are defined as logic formulas in FOKI, which is the main difference between FOKI and OWL. Zero-order logic is a mechanism used in FOKI to define alignment rules ([11]) and integration rules ([10]).

There are plenty of ontologies expressed in OWL, including those dedicated to ontology integration (e.g., OEAI benchmarks). To be able to process these ontologies in FOKI, a translation mechanism from OWL to FOKI meta-model is necessary. The translation must be two-directional, as the result of ontology integration should be comparable with existing benchmarks expressed in OWL. Translation FOKI-OWL is performed on the syntax level only. The semantics of particular elements (attributes, relations) are defined later by an expert.

To motivate our work and explain the source of the problems mentioned in the introduction, we use a simple example of ontology integration. Let's assume there are two ontologies, O1 and O2, to be integrated into OWL – see Table 1.

Each ontology introduces only one concept *Person* with a few attributes. These ontologies could be represented using our FOKI formal model as:

$O_1 = \{\ Person\ (name : string, birthdate : dateTime)\ \}$
$O_2 = \{Person\ (first\_name : string, last\_name : string,$
$age : nonNegativeInteger)\}$

For the ontologies' integration, the value of the S_A function has to be defined for each attribute. The semantics are later stored separately in CSV files. Having the semantics defined, the integration process can be run. The integration results could be, e.g. as follows:

$O_1\_O_2 = \{Person\ (name : string, birthdate : dateTime,$
$age : nonNegativeInteger)\ \}$

Now, to make the results readable, and useful for further processing, a transformation from FOKI to OWL is necessary. It could produce an ontology in the form given in Listing 1.1.

Table 1. Ontologies to be integrated in OWL format

O_1	O_2
<Declaration	<Declaration>
<Class IRI=”#Person”/>	<Class IRI=”#Person”/>
</Declaration>	</Declaration>
<Declaration>	<Declaration>
<DataProperty IRI=”#name”/>	<DataProperty IRI=”#first_name”/>
</Declaration>	</Declaration>
<DataPropertyDomain>	<Declaration>
<DataProperty IRI=”#name”/>	<DataProperty IRI=”#age”/>
<Class IRI=”#Person”/>	</Declaration>
</DataPropertyDomain>	<Declaration>
<DataPropertyRange>	<DataProperty IRI=”#last_name”/>
<DataProperty IRI=”#name”/>	</Declaration>
<Datatype	<DataPropertyDomain>
abbreviatedIRI=”xsd:string”/>	<DataProperty IRI=”#age”/>
</DataPropertyRange>	<Class IRI=”#Person”/>
<Declaration>	</DataPropertyDomain>
<DataProperty IRI=”#birthdate”/>	<DataPropertyDomain>
</Declaration>	<DataProperty IRI=”#first_name”/>
<DataPropertyDomain>	<Class IRI=”#Person”/>
<DataProperty IRI=”#birthdate”/>	</DataPropertyDomain>
<Class IRI=”#Person”/>	<DataPropertyDomain>
</DataPropertyDomain>	<DataProperty IRI=”#last_name”/>
<DataPropertyRange>	<Class IRI=”#Person”/>
<DataProperty IRI=”#birthdate”/>	</DataPropertyDomain>
<Datatype	<DataPropertyRange>
abbreviatedIRI=”xsd:dateTime”/>	<DataProperty IRI=”#age”/>
</DataPropertyRange>	<Datatype abbreviatedIRI
	=”xsd:nonNegativeInteger”/>
	</DataPropertyRange>
	<DataPropertyRange>
	<DataProperty IRI=”#first_name”/>
	<Datatype abbreviatedIRI=”xsd:string”/>
	</DataPropertyRange>
	<DataPropertyRange>
	<DataProperty IRI=”#last_name”/>
	<Datatype abbreviatedIRI=”xsd:string”/>
	</DataPropertyRange>

Listing 1.1. Integrated ontology

```
<Declaration><Class IRI="#Person"/></Declaration>
<Declaration><DataProperty IRI="#name"/></Declaration>
<Declaration><DataProperty IRI="#age"/></Declaration>
<Declaration><DataProperty IRI="#birthdate"/></Declaration>

<DataPropertyDomain>
<DataProperty IRI="#name"/><Class IRI="#Person"/>
</DataPropertyDomain>

<DataPropertyRange>
<DataProperty IRI="#name"/>
<Datatype abbreviatedIRI="xsd:string"/>
</DataPropertyRange>

<DataPropertyDomain>
<DataProperty IRI="#age"/><Class IRI="#Person"/>
</DataPropertyDomain>

<DataPropertyRange>
<DataProperty IRI="#age"/>
<Datatype abbreviatedIRI="xsd:nonNegativeInteger"/>
</DataPropertyRange>

<DataPropertyDomain>
<DataProperty IRI="#birthdate"/>
<Class IRI="#Person"/>
</DataPropertyDomain>

<DataPropertyRange>
<DataProperty IRI="#birthdate"/>
<Datatype abbreviatedIRI="xsd:dateTime"/>
</DataPropertyRange>
```

5. Transformation Procedure

This section presents detailed transformation procedures to convert ontologies between FOKI and the OWL2 standard. These two frameworks exhibit fundamental differences in their treatment of unknown facts. To bridge these differences, we designed a set of precise transformation rules to preserve the semantic integrity of the ontologies.

The intricacies of translating OWL's complex, expressive capabilities, particularly its data properties and logical constructs, into the formal structure of FOKI are explained. This includes handling equivalence, disjointness, and cardinality constraints in a closed-world context. Next, the rigorous testing and validation of our procedures using benchmark datasets provided by the Ontology Alignment Evaluation Initiative and the Ontology

Lookup Service is presented. The aim of our experimental research is to confirm the reliability and consistency of our transformation rules in practical ontology management applications.

5.1. Assumptions

The primary contribution of this paper is the development of transformation procedures that facilitate the conversion between two fundamentally distinct ontology representation notations: the newly developed FOKI formalism and the widely adopted OWL2 standard.

The critical difference between these frameworks lies in their approaches to managing unknown facts within a given domain. OWL2 operates under an open-world assumption, meaning that the absence of explicit information does not imply falsehood. In contrast, FOKI follows a closed-world assumption, where any statement not explicitly included in the ontology is considered false.

To bridge these conceptual differences effectively, we have established a detailed set of transformation rules. These rules are designed to be repeatable and concise, ensuring that the semantics of the input ontology are preserved as accurately as possible despite the inherent constraints of the differing frameworks. This transition involves careful consideration of OWL's non-functional data properties and open-world assumption, as well as FOKI's functional attributes and closed-world perspective. The meticulous design of these transformation rules aims to minimize information loss and maintain the integrity of the original ontology's semantics.

Furthermore, our approach addresses the challenges of converting OWL's rich, expressive capabilities into the more rigid structure required by FOKI. This includes handling equivalence, disjointness, and other logical constructs that may have different interpretations in a closed-world context. Our transformation procedures have been rigorously tested to ensure that they provide reliable and consistent results, facilitating the interoperability between systems using OWL2 and those based on the FOKI framework. This advancement not only supports seamless data integration across diverse ontological systems but also enhances the robustness and flexibility of ontology management practices.

The other difference relates to data properties. In OWL, an instance may have many valuations (literals) of a specific data property unless the property is marked as functional. In FOKI, an attribute (the equivalent of OWL data property) may have, at most, one value assigned independently on the case. No mechanism in FOKI allows combining attribute domains. It is why almost all OWL axioms referring to data properties (e.g., *DataUnionOf*, *DataComplementOf*) are not taken into consideration, including new data types declaration. Some other axioms, e.g., *DataMinCardinality*, are considered with specific restrictions (the cardinality must be equal to 1). At that moment, only OWL primitive types (declared in e.g., owl, RDF namespaces) are translated. For the unknown type, a special *anyURI* is assigned, which can be treated as a parent for all types.

Given the profound semantic differences between OWL2 and FOKI, achieving an utterly lossless transformation is impractical. Instead, our approach focuses on providing the best possible approximation by narrowing the transformation scope to the most frequently utilized elements in real-world ontologies. This decision was informed by extensive analysis of datasets provided by the Ontology Alignment Evaluation Initiative (OAEI) and the Ontology Lookup Service (OLS), which serve as benchmarks and repositories for state-of-the-art ontology applications and biomedical ontologies, respectively.

These platforms provide invaluable resources for validating and refining our transformation procedures, ensuring they meet the practical demands of ontology management and application in diverse domains. The former, OAEI (Ontology Alignment Evaluation Initiative), is a non-profit organization that has organized (since 2004) public campaigns aimed at evaluating ontology matching technologies. For this reason, it provides benchmark datasets with pre-prepared ontologies. These datasets are frequently treated as a state of the art benchmark data used to test various ontology-related applications. The latter, the Ontology Lookup Service (OLS), is a repository for biomedical ontologies that provides a single point of access to the latest ontology versions.

We tested more than 250 ontologies (first level only, without counting imported elements), and based on them, created a frequency distribution for all axioms. The obtained results are presented in Table 2. The table contains all axioms found in processed ontologies. Some axioms represent constructs that cannot be considered in transformations to FOKI formalism (e.g., annotations). These are written in italics. Out of the remaining axioms we have selected to implement in transformation rules those with at least 1% occurrence. As one can observe, the *DataPropertyRange* is the last axiom directly translated to FOKI (it represents an attribute domain) with 239 occurrences (more than 99.99% of axiom instances are above that limit).

5.2. FOKI Meta-model

Figures 1, 2, and 3 present the FOKI meta-model, which enables bi-directional translation between OWL2 and FOKI. Both OWL and FOKI ontologies can be represented as meta-model instances expressed in an abstract way (syntax agnostic).

The meta-model includes all elements specific to FOKI and those of OWL, which can be translated into FOKI. On one side, it is a limited version of the OWL2 meta-model, and – on the other side – it is an extension of the OWL2 meta-model as it contains elements presented in FOKI only (e.g., the semantics part). Because of its flexibility, we call it generic or universal.

The ontology, according to its formal definition (see chapter 3), is defined as a composition of concepts (C), generalization relationships between concepts (H), relations (R_C), instances of concepts (I), instances of relations (R_I), and predicates (Z) – see Figure 1. The predicates are specific to FOKI and are skipped in the meta-model. The other ontology components are directly visible in the meta-model, represented by associations and compositions written in blue.

There exists an apparent connection between OWL2 and the meta-model. The OWL2 class is represented by the *Concept* meta-class while the OWL2 instance – by the *Instance* meta-class.

OWL2 class is a container of data properties represented by the *Attribute* meta-class. Each attribute should have a domain defined. The domain is perceived as a set of primitive types defined in namespaces owl, RDF, XSD). Additionally, the attribute can be set as functional.

Each attribute is functional by default in FOKI (one can have at most one value assigned). Here, it is used to mark the fact that the translation OWL2-meta-model-FOKI can be potentially lossy (when *isFunctional* = false). However, instances are rarely defined in ontologies, and an attribute has extremely rarely more than one value (see Table 2).

Table 2. The frequency of the axioms occurring in ontologies provided by OAEI and OLS

Axioms' name	Count	Axioms' name	Count	Axioms' name	Count	Axioms' name	Count
[*AnnotationAssertion*]	11398600	[DataPropertyAssertion]	5637	[Declaration(DataProperty)]	530	[InverseFunctionalObjectProperty]	38
[SubClassOf]	2541198	[ObjectAllValuesFrom]	3920	[EquivalentObjectProperties]	419	[DisjointUnion]	33
[Declaration(Class)]	1725507	[ObjectHasValue]	3705	[ObjectOneOf]	379	[ReflexiveObjectProperty]	28
[ObjectPropertyAssertion]	1676802	[ObjectPropertyDomain]	3135	[DataPropertyRange]	239	[*DataIntersectionOf*]	23
[ObjectSomeValuesFrom]	1314347	[ObjectPropertyRange]	3068	[SymmetricObjectProperty]	230	[DisjointObjectProperties]	20
[Declaration(NamedIndividual)]	489422	[*SameIndividual*]	2721	[FunctionalObjectProperty]	152	[*DataAllValuesFrom*]	16
[ObjectIntersectionOf]	400322	[ObjectPropertyChain]	2391	[IrreflexiveObjectProperty]	106	[DataMaxCardinality]	14
[EquivalentClasses]	191466	[InverseObjectProperties]	2071	[*Declaration(Datatype)*]	103	[*AnnotationPropertyDomain*]	11
[ClassAssertion]	46979	[TransitiveObjectProperty]	1490	[*FunctionalDataProperty*]	82	[*DataOneOf*]	3
[*DataHasValue*]	14339	[*SubAnnotationPropertyOf*]	1462	[*SubDataPropertyOf*]	77	[*DatatypeDefinition*]	1
[ObjectExactCardinality]	12522	[*DataSomeValuesFrom*]	1196	[*AnnotationPropertyRange*]	75	[*DataUnionOf*]	1
[SubObjectPropertyOf]	11275	[*DatatypeRestriction*]	1191	[*DifferentIndividuals*]	71	[*DataComplementOf*]	0
[Declaration(ObjectProperty)]	11171	[ObjectMaxCardinality]	910	[AsymmetricObjectProperty]	67	[*EquivalentDataProperties*]	0
[*Declaration(AnnotationProperty)*]	9521	[ObjectComplementOf]	639	[ObjectHasSelf]	65	[*DisjointDataProperties*]	0
[DisjointClasses]	6916	[ObjectMinCardinality]	619	[DataExactCardinality]	58	[NegativeObjectPropertyAssertion]	0
[ObjectUnionOf]	6353	[ObjectInverseOf]	589	[DataMinCardinality]	40	[NegativeDataPropertyAssertion]	0

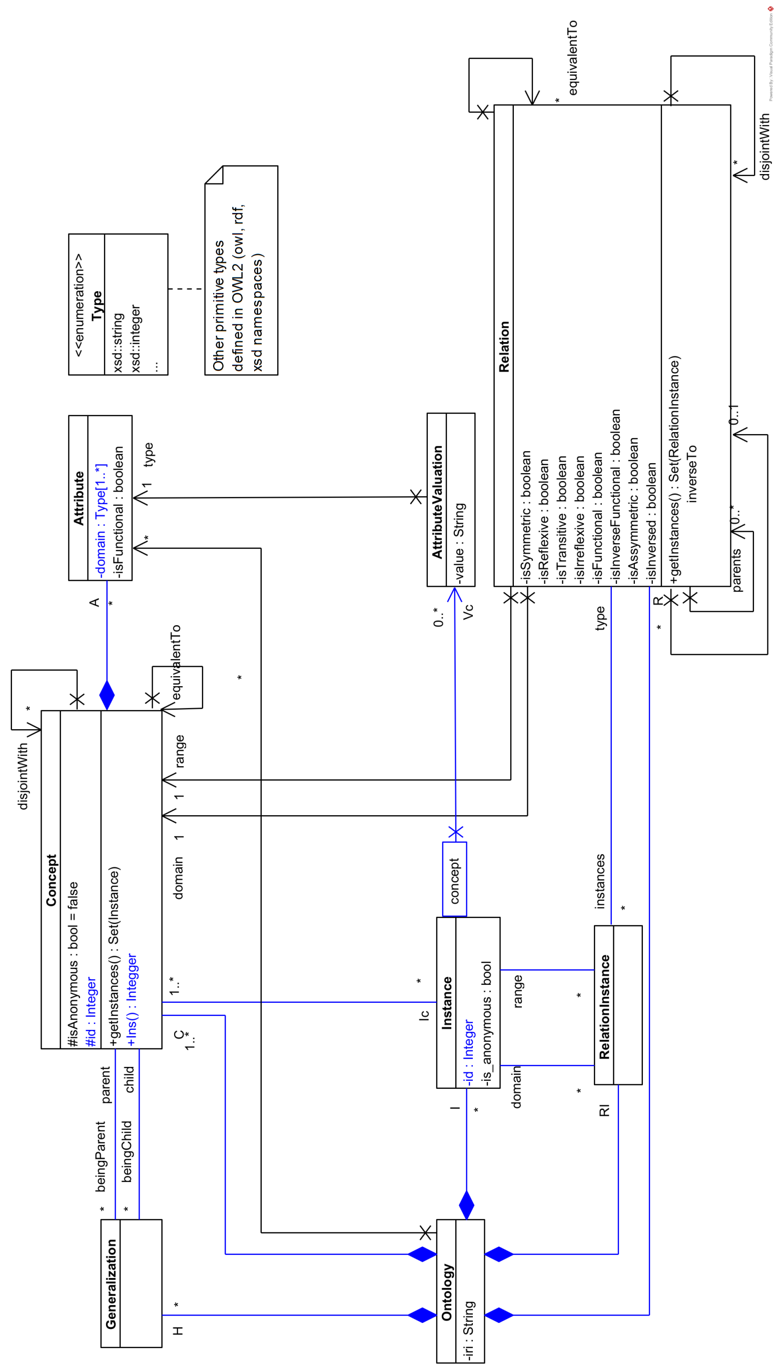

Fig. 1. FOKI meta-model – ontology basic structure

How OWL2 class expression axioms are represented in the meta-model depends strongly on its type. *SubClassOf* axiom is represented by *Generalization* meta-class, which directly links the parent and its child. Self-associations with the representative role names *equivalentTo, disjointWith* represent OWL *EquivalentClass* and *DisjointClass* axioms.

The meta-model *Relation* class represents an OWL2 object property. Such a relation can be marked as transitive, reflexive, symmetric, etc. It can also refer to its inverse version (a result of the translation of *InverseObjectProperty*). Essential axioms define the domain and range of a property. In the meta-model, both are represented by directional associations with specific roles (domain, range) from *Relation* to *Concept* meta-class. Similarly to classes, axioms about object properties equivalence, or the fact some are disjoint or inverse, are represented by directional self-associations with *Relation* meta-class on both ends. A relation can also store a list of its parents (if any).

The OWL2 concept instances are represented by the *Instance* meta-class. The qualifier *concept* represents this at the end of the association, linking the *Instance* meta-class with *AttributeValuation* meta-class (representation of the OWL2 Literal, e.g., 'John'). Similarly, instances of object properties are represented by the *RelationInstance* meta-class. The instance of the *Relation* meta-class remembers its features, i.e., if it is symmetric, reflexive, functional, etc.

Concepts and instances are named elements, i.e., they can be identified by an IRI (Internationalized Resource Identifier) from which the resource's name can be extracted (the name attribute) – see Figure 2. The name can include the package name, while its short version (the *shortName* attribute) is only limited to the resource name. For an anonymous instance, the *is_anonymous* attribute is set to false. As an instance can belong to many classes (concepts), their attributes can be split into groups (one group per concept).

Let us demonstrate the correspondence between the OWL ontology from Section 4 and our meta-model. In this example:

- O_1, O_2 are instances of the *Ontology* meta-class
- *Person* is an instance of the FOKI *Concept* meta-class (all the properties inherited from *NamedElement* meta-class, i.e. *name*, *iri* and *shortName* are set for *Person*, the concept is not anonymous); *Person* is a part of the proper ontology
- *name*, *birthdate* etc. are instances of the *Attribute* meta-class with properly set domain (*string* for *name*, *dateTime* for *birthdate*). The concept instance keeps attributes via composition relationship. The functional property for attributes is set to false.

Attributes (parts of *Concepts*) and *Relations* are elements to which semantics can be assigned in the form of a first-logic predicate sentence – see Figure 2. Such sentences are represented by instances of the *Semantics* class and its children (*Label, Operator*). Assuming that Person's *name* has semantics defined as $first\_name$ and $last\_name$, the semantics (associated with the *name* attribute) will be represented as two instances of the *Label* meta-classes, one for *first_name*, and one for *last_name*, connected by an instance of the *Operator* meta-class with the *type* property set to 'and'. The semantic part is not considered in the OWL2 to FOKI translation because this part is absent in OWL.

The core of every ontology is formed by a concept's definition and concept hierarchy – see Figure 3.

Any concept in OWL2 can be directly declared and have a name or be introduced indirectly as a combination of other concepts, e.g., union, intersection, etc. The combinations

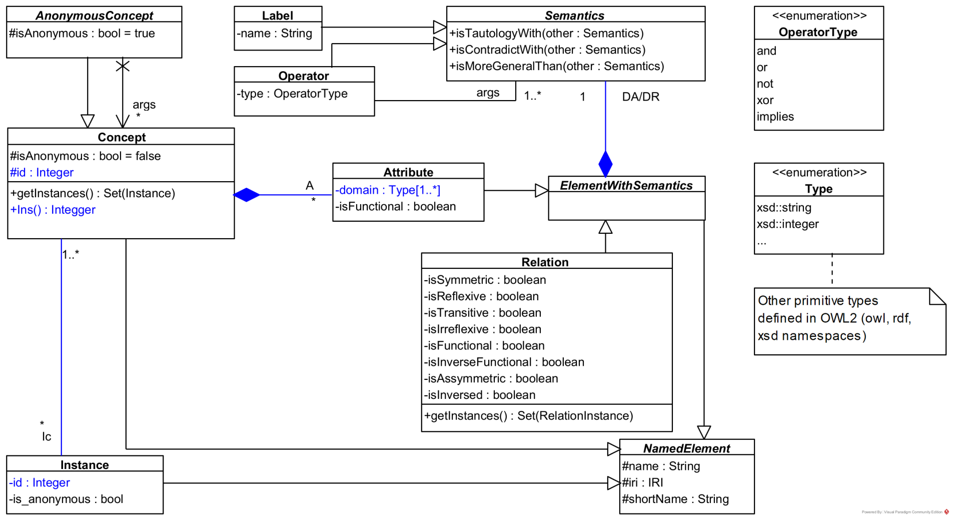

Fig. 2. FOKI meta-model – ontology named elements and elements with semantics

are represented as children of *AnonymousConcept* meta-class working on arguments (*args* role). The concrete children of *AnonymousConcepts* can be easily linked to the proper class expressions in OWL2, e.g. *ObjectIntersectionOf (ObjectIntersection)* in the meta-model), *ObjectUnionOf (ObjectUnion), ObjectComplementOf (ObjectComplement), ObjectSomeValuesFrom (ObjectValuesFrom, type*=some), *ObjectAllValuesFrom (ObjectValuesFrom, type*=all), *ObjectMinCardinality (ObjectCardinality, type* = min), *ObjectMaxCardinality (ObjectCardinality, type*=max), *ObjectExactCardinality (ObjectCardinality, type*=exact), *ObjectHasSelf*,*ObjectHasValue*, *ObjectOneOf*.

Such a solution enables the representation of complex structures, e.g., *UnionOf(IntersecionOf(A, B), C)* – see Figure 4. As one can notice, the names for anonymous concepts are created by the FOKI tool, and they reflect the intended semantics of the way the concept's arguments are combined.

DisjointUnion OWL axiom introduces a separate anonymous class – *DisjointUnion* in the meta-model.

5.3. Transformation Rules from OWL

The input ontology O, in OWL, can contain elements that are not explicitly defined. It means that some of its content must be inferred before any operations on reading ontology are possible. For the sake of consistency, the elements that can be inferred include:

– domain/ranges of relations if they are known for one of the parent relations
– instances of concepts/relations (a result of *getInstance* operation).

Examples of transformation procedures are presented below. The meta-model is a proxy between OWL2 and FOKI translation. Therefore, translation procedures are split

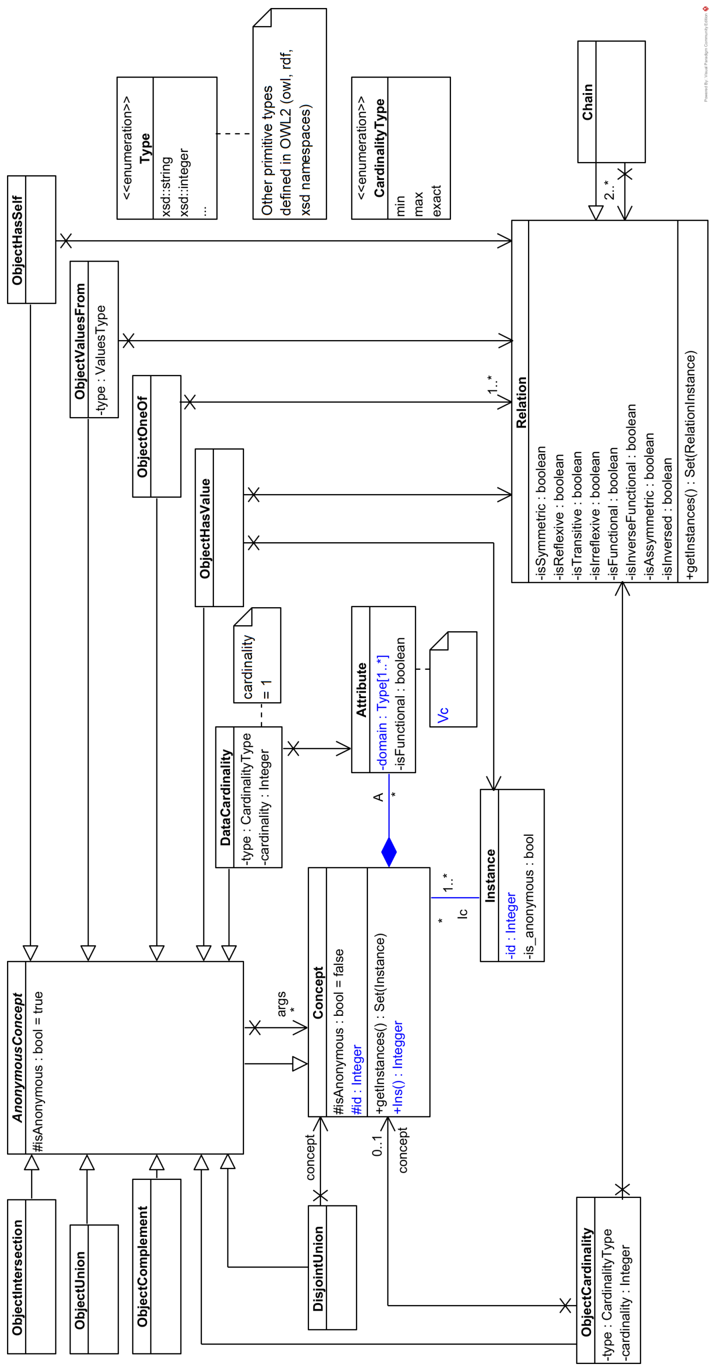

Fig. 3. FOKI meta-model – ontology concept heierarchy

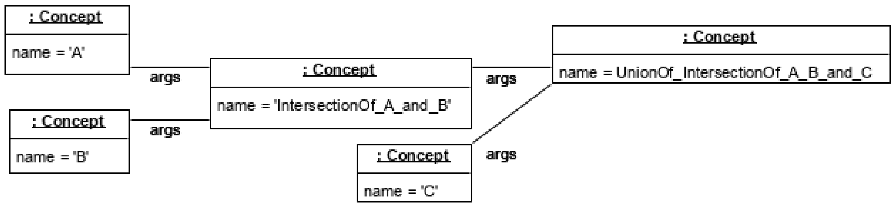

Fig. 4. Exemplary instance of a complex concept

into two groups. The first takes an OWL2 ontology and produces an instance of the meta-model. The second goes further and translates an instance of the meta-model into an instance of FOKI (the mathematical formalism). The opposite transformations, i.e., from FOKI formalisms to the instance of the meta-model and farther to the OWL are straightforward (defined indirectly by the existing transformation rules).

Let O_{OWL} represent an ontology expressed in OWL2, O_{MM} represent the same ontology as an instance of the meta-model, and O_{FOKI} represent the ontology O expressed in FOKI formalism (abstract syntax). These three representations are globally available for every formulated procedure. To select a specific element from one of the ontologies, a dot notation is used, e.g., $O_{FOKI}.C$ which represents the set of concepts C from the O_{FOKI} ontology. If the element belongs to the meta-model, the meta-class name proceeded by a '_' (to represent an instance) is used instead, e.g., *_Attribute*, *_Instance*, *_Relation*. The valuation of attributes for meta-class instances, if necessary, will be described with a dot notation, e.g. *_AttributeValuation.value* ='5'. In the same manner, the association ends will be referred, e.g., *_Ontology.I* = new *Set(Instance)* – where *I* is the association end representing a set of instances being part of the ontology.

Due to the limited space available for this paper, we present only some representatives of transformation procedures and divide them into two sections. The first shows how basic ontology elements are created. These elements are classes, attributes, relations between concepts, and relations between instances. The second demonstrates how relation properties are translated using a symmetry feature as an example. The last example concentrates on so-called *AnonymousConcepts*, i.e., concepts without a specific name constructed as a composition, e.g., a union or an intersection of other concepts.

OWL2 to meta-model translations The first transformation algorithm 1 shows how a new concept is created. When a new instance of a meta-model class is created, all its association ends are also initialized as empty collections (sets). For example, when an *Attribute* is created, then it is connected with an empty set of *AttributeValuations*. For simplicity, that fact is skipped in the transformation procedures.

The next four transformations 2, 3, 4, 5 present how axioms connected with *data property* are processed. The procedure 2 translates a data property declaration that introduces a data property name only. On that basis, a new attribute is created and stored (as a part of meta-model ontology) for further processing.

Algorithm 1 CreateConcept in meta-model.
OWL syntax: *Class(c)*

```
1: Input: $c \in O_{OWL}$
2: create new $\_Concept$
3: generate unique $id$
4: $\_Concept.id = id$
5: $\_Concept.name = c.name$
6: $\_Concept.IRI = c.IRI$
7: add $\_Concept$ to $O_{OWL}.\_Ontology.C$
```

Algorithm 2 CreateAttribute in meta-model.
OWL syntax: *Declaration(DataProperty(p))*

```
1: Input: $p(roperty) \in O_{OWL}$
2: create new $\_Attribute$
3: $\_Attribute.isFunctional = false$
4: $\_Attribute.name = p.name$
5: $\_Attribute.IRI = p.IRI$
6: add $\_Attribute$ to $O_{OWL}.\_Ontology.attribute$
```

The *FindOrCreateAttribute* function 3 is an auxiliary element, used later for translation of other axioms. The existence of similar functions, e.g. *FindOrCreateConcept*, *FindOrCreateRelation* etc., is assumed. These functions look for an element with a specific name. If it does not exist, the element is created.

Algorithm 3 FindOrCreateAttribute in meta-model

```
1: Input: $a, O_{OWL}$
2: Result: An attribute $\_Attribute$ – meta-model representation of $a$ in OWL2
3: if $O_{MM}.\_Ontology.attribute$ contains $\_Attribute$ : $\_Attribute.name = a.name$ then
4:     return $\_Attribute$
5: end if
6: return CreateAttribute($a, O_{OWL}$)
```

Translation of *DataPropertyDomain* axiom 4 results in finding or creation (if it does not exist) of an attribute that is assigned to the proper class (also found or created).

Translation of *DataPropertyRange* axiom 5 requires the proper attribute to be found (or created). After that, its feature *domains* are modified by adding a new range.

The procedure 6 shows the creation of complex concepts (by the example of *ObjectIntersectionOf*).

The next procedure 7 shows the creation of a concept instance, assuming that the instance is not an anonymous one (a result of the processing of *ClassAssertion* axiom).

Data property in OWL2 may have a value assigned to an instance that is a proper class member. In the meta-model, the data property value is represented by an instance of $AttributeValuation$ class. The instance uses a qualifier (concept) to split attribute values into manageable pieces (Algorithm 8).

Algorithm 4 Assign an attribute to a concept in meta-model
OWL syntax: *DataPropertyDomain(p, c)*

```
1: Input: $p(roperty), c(lass) \in O_{OWL}$
2: $\_Attribute$ = FindOrCreateAttribute($p, O_{MM}.\_Ontology$);
3: $\_Concept$ = FindOrCreateConcept($c, O_{MM}.\_Ontology$);
4: add $\_Attribute$ to $\_Concept.attribute$
```

Algorithm 5 Assign an attribute's domain in meta-model
OWL syntax: *DataPropertyRange(p, r)*

```
1: Input: $p(roperty), r(ange) \in O_{OWL}$
2: $\_Attribute$ = FindOrCreateAttribute($p, O_{MM}.\_Ontology$);
3: add $r$ to $\_Attribute.domains$
```

Algorithm 6 Create an anonymous concept
OWL syntax: *ObjectIntersectionOf*($c_1, c_2, \dots, c_n$)

```
1: Input: $c_1[OWL], c_2[OWL], \dots, c_n[OWL] \in O_{OWL}$
2: create new $\_ObjectIntersection$
3: add $\_ObjectIntersection$ to $O_{MM}.\_Ontology.concepts$
4: $\_ObjectIntersection.name = ObjectIntersectionOf\_ + c_1.name + \_and\_ + \dots + \_and\_ + c_n.name$
5: for $c_i \in \{c_1, \dots, c_n\}$ do
6:     add $c_i$ to $\_ObjectIntersection.args$
7: end for
```

Algorithm 7 Create an instance
OWL syntax: *ClassAssertion(c i)*

```
1: Input: $c(lass), i(nstance) \in O_{OWL}$
2: $\_Instance$ = FindOrCreateInstance($i, O_{MM}.\_Ontology$);
3: generate unique $id$;
4: $\_Instance.id = id$;
5: $\_Instance.is\_anonymous = false$;
6: $\_Concept$ = FindOrCreateConcept($c, O_{MM}.\_Ontology$);
7: add $\_Instance$ to $\_Concept.instances$;
8: add $\_Concept$ to $\_Instance.concept$;
```

Algorithm 8 Create an instance valuation
OWL syntax: *DataPropertyAssertion(p i v)*

```
1: Input: $p(roperty), i(nstance), v(alue) \in O_{OWL}$
2: create new $\_AttributeValuation$;
3: $\_AttributeValuation.value = v$;
4: $\_Instance$ = FindOrCreateInstance($i, O_{OWL}.\_Ontology$);
5: add $\_AttributeValuation$ to $\_Instance.attributes$;    ▷ Assuming $\_Instance[c]$ means to add to instance's attributes
```

Meta-model relation is created during the processing of *ObjectProperty* axioms (declarations, domain definition, range definition). These axioms are processed similarly to *DataProperty* axioms and therefore are skipped. Relation in OWL could have many features, e.g., be symmetric, reflexive, etc. The processing of *SymmetricObjectProperty* axiom is presented in Algorithm 9.

Algorithm 9 Create a symmetric relation
OWL syntax: *SymmetricObjectProperty(p)*

```
1: Input: $p(roperty) \in O_{OWL}$
2: $\_Relation = $ FindOrCreateRelation($p, O_{MM}.\_Ontology$);
3: $\_Relation.isSymmetric = $ true;
```

Transformation rules from meta-model to FOKI This section presents some procedures for translating elements taken from the presented meta-model into mathematical constructs defined according to the formal foundations presented in Section 3. Such transformation requires an auxiliary procedure, which takes as an input an instance for *Semantics* class from the meta-model. It recursively creates a logic sentence built from atomic literals and logic operators (negation, conjunction, disjunction, implication, and exclusive disjunction). This procedure is presented as Algoritm 10.

Algorithm 10 Create_Semantics

```
1: Input: $\_s \in O_{MM}.Semantics$
2: if $\_s$ is $O_{MM}.Label$ then
3:     return $\_s.name$;
4: else
5:     if $\_s.type = $ 'not' then
6:         return $\neg$ + Create_Semantics($\_s.args[0]$);
7:     else if $\_s.type = $ 'and' then
8:         return Create_Semantics($\_s.args[0]$) + $\wedge$ + Create_Semantics($\_s.args[1]$);
9:     else if $\_s.type = $ 'or' then
10:        return Create_Semantics($\_s.args[0]$) + $\vee$ + Create_Semantics($\_s.args[1]$);
11:    else if $\_s.type = $ 'xor' then
12:        return Create_Semantics($\_s.args[0]$) + $\oplus$ + Create_Semantics($\_s.args[1]$);
13:    else if $\_s.type = $ 'implies' then
14:        return Create_Semantics($\_s.args[0]$) + $\implies$ + Create_Semantics($\_s.args[1]$);
15:    end if
16: end if
```

The first transformation 11 shows how a concept's definition is built from an instance of a Concept. If the concept is anonymous, an additional predicate is stored in FOKI. The procedure also shows how the defined earlier procedure for creating logic sentences is used in the context of attributes' semantics.

Algorithm 11 Transformation of a concept

```
1: Input:$O_{MM}$, $\_c \in O_{MM}.C$
2: create new concept $c = (\_c.id, A^c = \emptyset, V^c = \emptyset, I^c = \emptyset)$;
3: for $\_a \in \_c.A$ do
4:     create an attribute $a$ named $\_a.name$;
5:     add $a$ to $A^c$;
6:     $O_{FOKI}.S_A(a, c)$ = Create_Semantics($\_a.DA/DR$);
7:     add $\_a.domain$ to $V^c$;
8: end for
9: add $c$ to $O_{FOKI}.C$;
10: if $\_c.isAnonymous$ then
11:     add $is\_anonymous(c)$ to $O_{FOKI}.Z$;
12: end if
```

The next procedure 12 addresses the transformation of instances. It allocates attributes' values to the particular instance created accordingly to the template enforced by a concept to which it will be eventually assigned. Similarly to the previous procedure (11), the situation in which a particular instance is not explicitly defined in OWL is handled by adding $is\_anonymous$ predicate.

Algorithm 12 Transformation of a concept's instance

```
1: Input: $O_{MM}$, $\_c \in O_{MM}.C$, $\_i \in O_{MM}.I \wedge \_i \in c.Ic$
2: create new instance $i = (\_i.id, v_c^i = \emptyset)$;
3: find $c \in O_{FOKI}.C$ such that $c.id = \_c.id$;
4: for $a \in \_i.V_c$ do
5:     add $a.value$ to $v_c^i$;
6: end for
7: $I^c = I^c \cup \{i\}$;
8: if $\_i.is\_anonymous$ then
9:     add $is\_anonymous(i.id)$ to $O_{FOKI}.Z$;
10: end if
```

The procedure 13 shows how a transformation of concepts hierarchy is performed. According to Equation 1 the generalization of concepts is expressed as an ordered pair of concepts included in the set *H*. Therefore, the transformation takes as input an instance of a *Generalization* class, which has two attributes pointing to two *Concept* class instances. Then, these concepts are extracted, the generalization representation is created, which is eventually added to the set *H* according to the mathematical definition.

Algorithm 13 Transformation of a concept's generalization

```
1: Input: $O_{MM}$, $\_h \in O_{MM}.H$
2: find $c_1 \in O_{FOKI}.C$ such that $\_h.parent.id = c_1.id$;
3: find $c_2 \in O_{FOKI}.C$ such that $\_h.child.id = c_2.id$;
4: add $(c_1, c_2)$ to $O_{FOKI}.H$;
```

The last two procedures (14, 15) focus on translating concepts and instances relations. The first ((Algorithm 14) addresses relations between concepts. It is a similar approach to the transformation of concept generalization, but it requires a *Relation* class instance as an input to extract participating concepts. It then utilizes a procedure for creating logic sentences to generate relation's semantics, which is eventually expanded with descriptions of particular relations features (e.g. its transitivity or reflexivity).

Algorithm 14 Transformation of a concept's relation

```
1: **Input:** $O_{MM}$, $\_r \in O_{MM}.R$
2: find $c_1 \in O_{FOKI}.C$ such that $\_r.domain.id = c_1.id$;
3: find $c_2 \in O_{FOKI}.C$ such that $\_r.range.id = c_2.id$;
4: find $r^C \in O_{FOKI}.R^C$ such that $r^C = \_r.name$;
5: add $(c_1, c_2)$ to $r^C$;
6: $O_{FOKI}.S_R(r) = Create\_Semantics(\_r.DA/DR)$;
7: if $\neg r.isSymmetric$ then
8:     concatenate $' \wedge is\_symmetric'$ to $S_R(r)$
9: end if
10: if $\neg r.isReflexive$ then
11:     concatenate $' \wedge is\_reflexive'$ to $S_R(r)$
12: end if
13: if $\neg r.isTransitive$ then
14:     concatenate $' \wedge is\_transitive'$ to $S_R(r)$
15: end if
16: if $\neg r.isIrreflexive$ then
17:     concatenate $' \wedge is\_irreflexive'$ to $S_R(r)$
18: end if
19: if $\neg r.isFunctional$ then
20:     concatenate $' \wedge is\_functional'$ to $S_R(r)$
21: end if
22: if $\neg r.isInverseFunctional$ then
23:     concatenate $' \wedge is\_inverse\_functional'$ to $S_R(r)$
24: end if
25: if $\neg r.isAssymetric$ then
26:     concatenate $' \wedge is\_assymetric'$ to $S_R(r)$
27: end if
```

The last procedure 15 performs a translation of relations between instances. It accepts an instance of *RelationInstance* class from the meta-model. Then it extracts two participating instances, which are eventually included in the appropriate relation taken from the set $O_{FOKI}.R^I$ according to Equation 1.

6. Evaluation Procedure

The meta-model and bi-directional transformation procedure were implemented in a prototype tool written in Java 1.8, and available in executable version (jar file) at [13].

The following testing procedure was used to check the correctness and completeness of the transformation process. After reading an input ontology, it was saved without

Algorithm 15 Transformation of instances relation

```
1: Input: $O_{MM}$, $\_r \in O_{MM}.RI$
2: find $r^I \in O_{FOKI}.R^I$ such that $r^I = \_r.name$;
3: find $i_1 \in O_{FOKI}.I$ such that $\_r.domain.id = i_1.id$;
4: find $i_2 \in O_{FOKI}.I$ such that $\_r.range.id = i_2.id$;
5: add $(i_1, i_2)$ to $r^I$
```

any change in the functional syntax for further comparison. Next, the input ontology was transformed into a meta-model instance. This instance was later saved under another name in the functional syntax. Such an approach made a textual comparison of two files possible (the original input ontology and the result of its processing) with generally available tools. We used the $diff$ tool that computes and displays the differences between the contents of two files. When any textual difference was identified, it was checked if the FOKI translation was semantically different from the original version.

The testing procedure was run for commonly available ontologies of different sizes and natures. The results are presented in Table 3. The first column contains information about the ontology processed, the second – about the number of axioms and annotations (not translated) in the ontology, the third – about not translated axioms, and the fourth some notes/explanations of the results.

In most places when syntax differences were identified, the semantics were preserved, e.g. *InverseObjectProperties*(:forEvent :hasCall) is an equivalent for *InverseObjectProperties* (:hasCall :forEvent)). Once it occurred that the original ontology was redundant – some axioms could be eliminated without changing the semantics (they can be inferred from existing axioms). The translation didn't contain the redundant axioms. And once, the axiom *InverseObjectProperties*(:parallel_with :parallel_with) is replaced with the equivalent one *ReflexiveObjectProperty*(:parallel_with).

7. Future Works and Summary

In this paper, we introduced a meta-model and a comprehensive set of rules designed to facilitate the translation of ontologies between the FOKI framework's internal mathematical formalism and the widely used OWL2 standard. Utilizing UML notation, we meticulously defined the meta-model and outlined transformation procedures that effectively bridge these distinct frameworks. Each transformation procedure was crafted to align an abstract syntax element of OWL2 with its corresponding representation in the FOKI framework using specific meta-model elements.

The meta-model and the bi-directional transformation procedures, a result of our research, were implemented in Java. This practical implementation allowed us to rigorously test our translation method against ontologies primarily sourced from the Ontology Alignment Evaluation Initiative. These ontologies, serving as a benchmark dataset, are essential for validating many ontology-related tools.

The input ontology was initially preserved in its original functional syntax as a baseline for subsequent comparisons. It was then transformed into a meta-model instance and saved under a new identifier in the functional syntax. This approach facilitated a detailed textual comparison between the original input ontology and its processed counterpart

Table 3. The results of transformation

Ontology name and source	**Axiom number Annotation number**	**Percentage of correctly transformed axioms**	**Not transformed axioms or transformed with some changes**	**Comments**
Shop	92/0	92 (100%)	-	The FOKI translation contains one additional axiom: Declaration(Class(owl:Thing))
EDAS[2]	194/0	193 (99.9%)	DifferentIndividuals (1)	
Sigkdd [2]	892/11	891 (99.9%)		ObjectPropertyRange(:award :Award) – redundancy
ConfOf [2]	270/65	269 (99.9%)		Instead of: InverseObjectProperties(:parallel_with :parallel_with) It is: ReflexiveObjectProperty(:parallel_with)
Conference [2]	408/0	399 (97.7%)	Declaration(Datatype(xsd:date)) (1) Usage of datatype (8)	Instead of: DataPropertyRange(:is_an_ending_date xsd:date) It is: DataPropertyRange(:is_an_ending_date xsd:anyURI)
Movie Ontology [3]	870/0	861 (98.9%)	DifferentIndividuals (6) Declaration(Datatype(xsd:date)) (1) Usage of datatype (1)	Declaration(Datatype(xsd:date)) ObjectPropertyDomain (movieontology:hasActress www:Movie) ObjectPropertyDomain (movieontology:hasMaleActor www:Movie) – axioms can be inferred from definition of hasActor object property.
Sabine_target [4]	2896/784	2894 (99.9%)	DataPropertyAssertion (s///abine:hasTopicID :EU ”130”8sd:int) DataPropertyAssertion(sabine:hasTopicID :Politica_sociale ”300”8sd:int) (2)	In FOKI all attributes are functional by definition, and it is why only the last value assigment is remembered.
Sabine _source [4]	2582/784	2580 (99.9%)	DataPropertyAssertion (sabine:hasTopicID :EU ”164”8sd:int) DataPropertyAssertion(sabine:hasTopicID :Immigration ”281”8sd:int) (2)	In FOKI all attributes are functional by definition, and it is why only the last value assigment is remembered.
Human [5]	14857/15507	14854 (99.9%)	Declaration (AnnotationProperty(...)) (3)	.
Mouse [5]	7592/3451	7588 (99.9%)	Declaration (AnnotationProperty(...)) (4)	.

using standard tools. The transformation accuracy, a key measure of our methodology, consistently exceeded 95% across all tested ontologies. The minor losses observed were primarily due to inherent differences between the OWL and FOKI frameworks. Despite this, the results were highly satisfactory, demonstrating that OWL2 ontologies can now be effectively processed using the FOKI framework. Non-functional attributes are adeptly managed within the transformation, which is primarily lossy only in rare cases where an attribute is assigned multiple values. Addressing this issue could be achieved by enhancing the complexity of the transformation, possibly by representing a non-functional attribute domain as a collection in FOKI.

In the future, our efforts will concentrate on developing a concrete syntax and a query language for FOKI, as currently, the framework is equipped only with an abstract syntax. For FOKI to be utilized in practical applications, it is imperative to either adapt an existing ontology syntax or devise a new one tailored to its unique requirements.

References

1. Afzal, H., Waqas, M., Naz, T.: Owlmap: fully automatic mapping of ontology into relational database schema. International journal of advanced computer science and applications 7(11) (2016)
2. Andon, P., Reznichenko, V., Chistyakova, I.: Mapping of description logic to the relational data model. Cybernetics and Systems Analysis 53, 963–977 (2017)
3. Astrova, I., Korda, N., Kalja, A.: Storing owl ontologies in sql relational databases. International Journal of Electrical, Computer and Systems Engineering 1(4), 242–247 (2007)
4. Athanasiadis, I.N., Villa, F., Rizzoli, A.E.: Ontologies, javabeans and relational databases for enabling semantic programming. In: 31st Annual International Computer Software and Applications Conference (COMPSAC 2007). vol. 2, pp. 341–346. IEEE (2007)
5. Baader, F.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
6. Caldarola, E.G., Picariello, A., Rinaldi, A.M.: An approach to ontology integration for ontology reuse in knowledge based digital ecosystems. In: Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems. pp. 1–8 (2015)
7. De Paepe, D., Thijs, G., Buyle, R., Verborgh, R., Mannens, E.: Automated uml-based ontology generation in oslo 2. In: The Semantic Web: ESWC 2017 Satellite Events: ESWC 2017 Satellite Events, Portorož, Slovenia, May 28–June 1, 2017, Revised Selected Papers 14. pp. 93–97. Springer (2017)
8. El Hajjamy, O., Alaoui, K., Alaoui, L., Bahaj, M.: Mapping uml to owl2 ontology. Journal of Theoretical and Applied Information Technology 90(1), 126 (2016)
9. Hnatkowska, B.: Automatic sumo to uml translation. e-Informatica Software Engineering Journal 10(1) (2016)
10. Hnatkowska, B., Kozierkiewicz, A., Pietranik, M.: Semi-automatic definition of attribute semantics for the purpose of ontology integration. IEEE Access 8, 107272–107284 (2020)
11. Hnatkowska, B., Kozierkiewicz, A., Pietranik, M.: Fuzzy based approach to ontology relations alignment. In: 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–7. IEEE (2021)
12. Hnatkowska, B., Woroniecki, P.: Transformation of owl2 property axioms to groovy. In: SOFSEM 2018: Theory and Practice of Computer Science: 44th International Conference on Current Trends in Theory and Practice of Computer Science, Krems, Austria, January 29-February 2, 2018, Proceedings 44. pp. 269–282. Springer (2018)

13. Hnatkowska, B.: Owl2 to foki translator. `https://github.com/bhnatkowska/OWL2FOKI` (2020)
14. Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J.: Bootox: Practical mapping of rdbs to owl 2. In: The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II 14. pp. 113–132. Springer (2015)
15. Karpovic, J., Nemuraite, L., Stankeviciene, M.: Requirements for semantic business vocabularies and rules for transforming them into consistent owl2 ontologies. In: Information and Software Technologies: 18th International Conference, ICIST 2012, Kaunas, Lithuania, September 13-14, 2012. Proceedings 18. pp. 420–435. Springer (2012)
16. Kendall, E., Linehan, M.H.: Mapping sbvr to owl2. Tech. rep., IBM Research Report, RC25363 (WAT1303-040) March (2013)
17. Kozierkiewicz, A., Pietranik, M.: The knowledge increase estimation framework for integration of ontology instances' relations. In: Databases and Information Systems: 13th International Baltic Conference, DB&IS 2018, Trakai, Lithuania, July 1-4, 2018, Proceedings 13. pp. 172–186. Springer (2018)
18. Kozierkiewicz-Hetmańska, A., Pietranik, M.: The knowledge increase estimation framework for ontology integration on the concept level. Journal of Intelligent & Fuzzy Systems 32(2), 1161–1172 (2017)
19. Kozierkiewicz-Hetmańska, A., Pietranik, M., Hnatkowska, B.: The knowledge increase estimation framework for ontology integration on the instance level. In: Intelligent Information and Database Systems: 9th Asian Conference, ACIIDS 2017, Kanazawa, Japan, April 3-5, 2017, Proceedings, Part I 9. pp. 3–12. Springer (2017)
20. Maree, M., Belkhatir, M.: Addressing semantic heterogeneity through multiple knowledge base assisted merging of domain-specific ontologies. Knowledge-Based Systems 73, 199–211 (2015)
21. Niles, I., Pease, A.: Towards a standard upper ontology. In: Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001. pp. 2–9 (2001)
22. Osman, I., Yahia, S.B., Diallo, G.: Ontology integration: approaches and challenging issues. Information Fusion 71, 38–63 (2021)
23. Pietranik, M., Nguyen, N.T.: A multi-attribute based framework for ontology aligning. Neurocomputing 146, 276–290 (2014)
24. Stoilos, G., Geleta, D., Shamdasani, J., Khodadadi, M.: A novel approach and practical algorithms for ontology integration. In: The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part I 17. pp. 458–476. Springer (2018)
25. Tirmizi, S.H., Aitken, S., Moreira, D.A., Mungall, C., Sequeda, J., Shah, N.H., Miranker, D.P.: Mapping between the obo and owl ontology languages. Journal of biomedical semantics 2(1), 1–16 (2011)
26. Vysniauskas, E., Nemuraite, L.: Transforming ontology representation from owl to relational database. Information technology and control 35(3) (2006)
27. Xu, Z., Ni, Y., He, W., Lin, L., Yan, Q.: Automatic extraction of owl ontologies from uml class diagrams: a semantics-preserving approach. World Wide Web 15, 517–545 (2012)
28. Zedlitz, J., Jörke, J., Luttenberger, N.: From uml to owl 2. In: Knowledge Technology Week, pp. 154–163. Springer (2011)
29. Zina, N., Kaouther, N.: Automatically building database from biomedical ontology. In: International Work-Conference on Bioinformatics and Biomedical Engineering. pp. 1403–1411 (2014)

Adrianna Kozierkiewicz received the M.Sc. degree in computer science and the Ph.D. degree in 2007 and 2011, respectively. She is currently an associate professor at the Faculty of Information and Communication Technology at Wrocław University of Science and Technology. She is the author or co-author of over 80 scientific publications, including articles in journals from the Journal Citation Report (JCR) list. She is also the editor of 4 books published by the prestigious Springer publishing house. Her research interests focus on knowledge processing techniques and artificial intelligence, which are integral to modern computer systems. Her research work has been funded by the National Centre for Research and Development or the National Science Centre. She also conducts her research in cooperation with multiple business partners. In addition, she is the Editor in Chief of the International Journal of Intelligent Information and Database System and a member of the Computational Collective Intelligence research group - IEEE SMC. For several years, she has been involved in the organization of international scientific conferences, including ICCCI: International Conference on Computational Collective Intelligence and ACIIDS: Asian Conference on Intelligent Information and Database Systems. She has also organized several special sessions and given numerous presentations at international conferences, including keynote lectures.

Marcin Pietranik is an assistant professor at Wrocław University of Science and Technology since 2016. He obtained his M.Sc and Ph.D. degrees in computer science in 2008 and 2014, respectively. Dr. Pietranik is passionate about knowledge integration and is interested in ontology management, particularly ontology evolution and alignment. He has published more than 45 articles and has co-organized several conferences, demonstrating his significant contribution to the field. In addition to his academic pursuits, Dr. Pietranik has a practical background in modern web technologies, especially in the context of medical and telecommunication projects, which provides him with a unique perspective in his research.

Bogumiła Hnatkowska received an M.Sc. and Ph.D. in computer science from the Wrocław University of Science and Technology, Poland, in 1992 and 1997, respectively. Her Ph.D. dissertation was associated with using formal methods in software engineering. Since 2024, she has worked as a Professor at the Faculty of Information and Communication Technology. She has over 120 publications in international journals and conference proceedings from different areas of software engineering. Her main scientific interests include but are not limited to software development processes, modeling languages, model-driven development (also with the use of ontologies), and the quality of the software products. She is a member of program committees at several international conferences.

Received: February 27, 2024; Accepted: August 29, 2024.