A UNIFYING FORMAL ONTOLOGY MODEL
A Simple Formal Model for Unifying the Presentation of Ontologies in Semantic Web Research

Stephan Grimm
FZI – Research Center for Information Technology at the University of Karlsruhe
Haid-Und-Neu-Str, 10-14, 76131 Karlsruhe, Germany

Keywords: Ontologies, Formal Model, Semantic Web.

Abstract: We propose a simple formal ontology model for the uniform presentation of ontologies across different areas of SemanticWeb (SW) research. On the one hand, the models simplicity allows for abstracting from technical details in a selective way, while on the other hand it captures the essential characteristics of an ontology common to most ontology languages and formalisms. To demonstrate the compatibility of our model to existing SW research, we provide mappings to several languages and formalisations ranging from expressive language standards to light-weight semantical models. Moreover, we specify an extension of the UML metamodel for the graphical visualisation of ontologies, and we sketch a software architecture for the programmatic access to ontologies in terms of an API.

1 INTRODUCTION

Ontologies play a key role in the *Semantic Web* (SW) as conceptual yet computational models for the explicit representation of domain knowledge in information systems. One particular aspect of ontologies as computational knowledge representation artifacts is their *formality* according to a well-known definition in (Gruber, 1993), which allows for their automated processing by machines in a meaningful and clearly defined way based on the principles of formal semantics. In various areas that influence the SW, such as knowledge representation and reasoning, ontology alignment or Web 2.0 and light-weight semantics, however, the formal aspects of ontologies are perceived in different ways, emphasising different characteristics. In consequence, formal models used to describe ontologies in these areas deviate from each other and each sub-community has its own formal definition of an ontology.

To mediate between the various areas in the SW community, striving for synergy in different strands of ontology research, it would be beneficial to have a common basis for a formal model of an ontology that provides a unified view on the subject across different perspectives of processing techniques and use cases. Such a common formal model should, on the one hand, be powerful and expressive enough to capture all the specific characteristics of an ontology across the various fields. On the other hand, such a model should not be overly formal in cases where certain aspects are to be abstracted from, providing a simple view that is easily and intuitively understood by researchers from all areas.

Besides the unifying framework for the technical formalities of an ontology that such a model provides, there are aspects of usage towards graphical visualisation of and programmatic access to ontologies.

In this paper, we propose a unifying formal model for the presentation and investigation of ontologies across SW research strands. We demonstrate its compatibility to current SW research by providing mappings to existing ontology languages and formalisations in different SW areas. Moreover, we sketch a UML metamodel extension for graphical visualisation of and a software API for programmatic access to ontologies, both in accordance with our formal model.

2 ONTOLOGY ESSENTIALS

According to the well-established definition in (Gruber, 1993), an ontology is a "*formal explicit specification of a shared conceptualisation of a domain of interest*". Hence, it combines various aspects, such

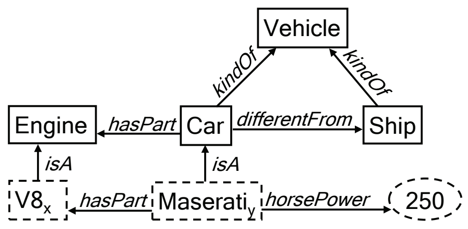

Figure 1: An example ontology as semantic network.

as *formality* and *explicitness* of represented knowledge for machine-processing, *consensus* for community agreement, and *conceptuality* and *domain specificity* for the capturing of domain knowledge in form of a conceptual model. In contrast to other conceptual models used in computer science, such as UML diagrams of software systems or entity-relationship diagrams for data storage, the primary purpose of an ontology is to serve as a source of domain knowledge to be queried by an information system that bases runtime decisions on the respective answers.

Ofttimes, ontologies are visualised and thought of as *semantic networks* that display interrelated conceptual nodes, as exemplarily depicted in Figure 1.

2.1 Essential Characteristics

Although there are various ontology languages with different semantics for similar language constructs next to a multitude of differing ontology formalisations, we can identify some *essential characteristics* that seem to be common to all presentations of ontologies across different research areas.

- *interrelation* – Without any interconnection, the plain nodes in a semantic network like the one depicted in Figure 1 would rather be a list of keywords. *Relations* between concepts allow for such an interconnection, enriching the conceptual nodes with structure. In the example, cars are related to engines by means of a *hasPart* connection.
- *instantiation* – An essential distinction is that made between concrete individual objects of the domain of interest and more general categories that group together objects with some common characteristics. *Instantiation* is the mechanism to allow for this distinction by assigning individual objects to classes as their instances, such as the specific Maserati in Figure1 is an instance of the class of all cars. This mechanism can be found across almost all forms of conceptual models, sometimes even in the form of metamodelling (Motik, 2005) allowing for classes to be instances of (meta-) classes themselves.
- *subsumption* – The most common way of interlinking general conceptual nodes is by *subsumption*, expressing a *kindOf*-relationship that reflects the notion of specialisation/generalisation. In the example, cars are stated to be kinds of vehicles, and thus, inherit their properties being their specialisation. Subsumption is the mechanism behind the feature of inheritance hierarchies prevalent in conceptual modelling.
- *exclusion* – A rather sophisticated means of representation is to state "negative" knowledge in form of class *exclusion*, preventing two general conceptual nodes from overlapping in their extensions. In the example, cars are stated to be different from ships, meaning that being a car excludes being a ship. This form of negative knowledge is found in rather expressive ontology languages that allow for negation.
- *axiomatisation* – The interrelated conceptual nodes as such are often not sufficient to express rich knowledge – they need to take part in complex statements about the domain of interest, which accounts for the notion of *axiomatisation*. Besides subsumption, exclusion or instantiation as simple forms of axiomatisation, many ontology languages allow for the formulation of more complex statements in form of general *axioms*. Complex axioms are typically neglected in the graphical presentation of an ontology in lack of an appropriate means for visualisation.
- *attribution* – Although not motivated by the underlying logical formalisms for knowledge representation, the inclusion of statements about datatypes and their values are a common feature in ontology languages, which accounts for the *attribution* of conceptual nodes by strings, numbers, etc. In the example, cars are attributed with a number indicating their horsepower. Datatypes and values are often an indispensable feature in many applications of ontologies.

2.2 Model Design Goals

Our primary goal is to design a formal ontology model that is on the one hand simple to suit a concise presentation of ontologies, and on the other hand rich enough to account for all the essential characteristics identified above. It should allow for emphasising any of these characteristics while abstracting from any of the others. In particular, it should be possible to either take an axiomatic view on an ontology not being concerned with structural aspects, or to take a graph-like view not mixing in logic-based axiomatisation. Despite all abstraction, the model should be compatible to the common ontology languages and formalisms for cases in which details of formal semantics and other knowledge representation specifics are to be discussed.

As the main use of such a unifying formal model we see the possibility of presenting ontology-based work in the course of Semantic Web research in a uniform way across different areas. Once such a model is established, however, it can facilitate more far-ranging uniform handling of ontologies. On the one hand, it can form the basis for a uniform graphical visualisation of ontologies that accounts for the above characteristics, e.g. in the widely accepted UML standard. On the other hand, it can serve as a structural framework for a uniform programmatic access to ontologies in software systems in form of an API layered on top of existing ontology software back-ends, e.g. in the widely used Java programming language.

3 A FORMAL MODEL FOR ONTOLOGIES

In this section, we present our formal ontology model as such, describing the constituents of an ontology as well as a set of conditions that determine their characteristics. Moreover, we report on some design decisions that have influenced the model.

3.1 Constituents of an Ontology

An *ontology* $\mathcal{O}$ is a tuple

$$\mathcal{O} = (\mathbf{S}, \mathbf{A}) \qquad (1)$$

of a *signature* **S** and a set of *axioms* **A**. The signature comprises several sets

$$\mathbf{S} = \mathsf{C} \cup \mathsf{I} \cup \mathsf{P} \qquad (2)$$

of *classes* C, *instances* I and *properties* P. They are further divided in sets

$$\mathsf{C} = \mathcal{C} \cup \mathcal{D}\ ,\ \mathsf{I} = \mathcal{I} \cup \mathcal{V}\ ,\ \mathsf{P} = \mathcal{R} \cup \mathcal{T} \qquad (3)$$

of *concepts* $\mathcal{C}$ and *data types* $\mathcal{D}$, *individuals* $\mathcal{I}$ and *data values* $\mathcal{V}$, as well as *relations* $\mathcal{R}$ and *attributes* $\mathcal{T}$. We also call the elements of the signature the *entities* of an ontology.

An axiom $\alpha \in \mathbf{A}$ is an expression in a first-order logical language over the signature **S** with the following restrictions:

- any entity $E \in \mathbf{S}$ occurs in some $\alpha \in \mathbf{A}$
- predicate symbols in α are taken from $\mathsf{C} \cup \mathsf{P}$, where symbols taken from C must be used as unary predicates in α
- constant symbols in α are taken from I
- any $t \in \mathcal{T}$ must be used as a binary predicate $t(i, v)$ with $i \in \mathcal{I}$ and $v \in \mathcal{V}$ (or i, v first-order variables)

Based on the essential characteristics from Section 2, we identify some special types of axioms that are common to most ontology languages and formalisations, for which we introduce a special notation in Table 1. Their intuitive meaning is as follows.

- *instantiation* – An instantiation axiom assigns an instance to a class.
- *assertion* – An assertion axiom assigns two instances by means of a property.
- *subsumption* – A subsumption axiom for two classes states that any instance of any one class is also an instance of the other class, while for two properties it states that any two instances connected by the one property are also connected by the other one.
- *domain* – A domain axiom for a property and a class states that for any connection of two instances by that property the source element is an instance of the domain class.
- *range* – A range axiom for a property and a class states that for any connection of two instances by that property the target element is an instance of the range class.
- *disjointness* – A disjointness axiom for two classes states that no instance of the one class can also be an instance of the other class, and thus, the classes exclude each other.

3.2 Normative and Semantic Conditions

We define some conditions that can be employed to further restrict the use of an ontology's entities and axioms. We divide them into *normative conditions* for restricting the signature and *semantic conditions* for restricting the semantics of axioms.

3.2.1 Normative Conditions

Some normative conditions that reflect basic aspects of formalisms for conceptual models are listed in Table 2. By separating axioms from entities, N_1 prevents reification of the form of axioms about entities that are axioms themselves, such as statements about triples in RDF (Klyne and Carroll, 2004). Moreover, N_2 prevents any form of metamodelling by clearly separating instances, classes and properties (no class can be used in place of an instance). Furthermore, N_3 separates abstract domain objects from concrete data

Table 1: Special axioms common in ontology formalisms.

Axiom Type	*Notation*	*first-order expression*
instantiation	$\alpha_{\wedge}(i,C)$	$[C(i)],\ i \in \mathsf{I}, C \in \mathsf{C}$
assertion	$\alpha_{\rightarrow}(i_1,p,i_2)$	$[p(i_1,i_2)],\ i_1,i_2 \in \mathsf{I}, p \in \mathsf{P}$
subsumption	$\alpha_{\triangle}(E_1,E_2)$	$[\forall x : E_1(x) \rightarrow E_2(x)],\ E_1,E_2 \in \mathsf{C} \cup \mathsf{P}$
domain	$\alpha_{D\rightarrow}(p,D)$	$[\forall x,y : p(x,y) \rightarrow D(x)],\ p \in \mathsf{P}, D \in \mathsf{C}$
range	$\alpha_{\rightarrow R}(p,R)$	$[\forall x,y : p(x,y) \rightarrow R(y)],\ p \in \mathsf{P}, R \in \mathsf{C}$
disjointness	$\alpha_{\oplus}(C_1,C_2)$	$[\forall x : C_1(x) \wedge C_2(x) \rightarrow \perp],\ C_1,C_2 \in \mathsf{C}$

Table 2: Normative conditions.

ID	*Name*	*Condition(s)*
N_1	non-reifiability	$\mathbf{A} \cap \mathbf{S} = \varnothing$
N_2	non-metalayering	a) $\mathsf{I} \cap \mathsf{C} = \varnothing$ b) $\mathsf{I} \cap \mathsf{P} = \varnothing$ c) $\mathsf{C} \cap \mathsf{P} = \varnothing$ d) $\forall r \in \mathcal{R} : \mathsf{arity}(r) > 1$
N_3	data-separation	a) $\mathcal{C} \cap \mathcal{D} = \varnothing$ b) $\mathcal{I} \cap \mathcal{V} = \varnothing$ c) $\mathcal{R} \cap \mathcal{T} = \varnothing$
N_4	relation-binarity	$\forall r \in \mathcal{R} : \mathsf{arity}(r) = 2$

N_5	non-instantiation	$\forall \alpha \in \mathbf{A}$:
		for $\alpha = [\ldots p(t) \ldots]$,
		if $p \in \mathsf{C}$ then $t \notin \mathsf{I}$

values and their types, as e.g. done in OWL (Patel-Schneider et al., 2004). By N_4, only binary relations are allowed, and N_5 prevents the instantiation of classes by restricting terms of class predicates in axioms to not be constants.

3.2.2 Semantic Conditions

The notion of *implicit knowledge* associated with an ontology $\mathcal{O}$ in terms of *deduction* is reflected in our model by the symbol $\langle \mathbf{A} \rangle$, which denotes the deductive closure of the set $\mathbf{A}$ of axioms in $\mathcal{O}$. Although different ontology languages and knowledge representation formalisms exhibit different semantics that yield varying sets $\langle \mathbf{A} \rangle$ for an ontology, we identify some semantic conditions that hold for most of the commonly used semantics, regardless of the underlying ontology language or model. These conditions characterise the "meaning" of the specific axioms outlined in Table 1 and are listed in the following.

S_1: (superclass instantiation) if $\alpha_{\wedge}(i,C_1) \in \langle \mathbf{A} \rangle$ and $\alpha_{\triangle}(C_1,C_2) \in \langle \mathbf{A} \rangle$ then $\alpha_{\wedge}(i,C_2) \in \langle \mathbf{A} \rangle$

S_2: (superproperty instantiation) if $\alpha_{\rightarrow}(i_1,p_1,i_2) \in \langle \mathbf{A} \rangle$ and $\alpha_{\triangle}(p_1,p_2) \in \langle \mathbf{A} \rangle$ then $\alpha_{\rightarrow}(i_1,p_2,i_2) \in \langle \mathbf{A} \rangle$

S_3: (subsumption transitivity) if $\alpha_{\triangle}(E_1,E_2) \in \langle \mathbf{A} \rangle$ and $\alpha_{\triangle}(E_2,E_3) \in \langle \mathbf{A} \rangle$ then $\alpha_{\triangle}(E_1,E_3) \in \langle \mathbf{A} \rangle$

S_4: (property domain) if $\alpha_{D\rightarrow}(p,D) \in \langle \mathbf{A} \rangle$ and $\alpha_{\rightarrow}(i_1,p,i_2) \in \langle \mathbf{A} \rangle$ then $\alpha_{\wedge}(i_1,D) \in \langle \mathbf{A} \rangle$

S_5: (property range) if $\alpha_{\rightarrow R}(p,R) \in \langle \mathbf{A} \rangle$ and $\alpha_{\rightarrow}(i_1,p,i_2) \in \langle \mathbf{A} \rangle$ then $\alpha_{\wedge}(i_2,R) \in \langle \mathbf{A} \rangle$

S_6: (disjointness) if $\alpha_{\oplus}(C_1,C_2) \in \langle \mathbf{A} \rangle$ then $\alpha_{\oplus}(C_2,C_1) \in \langle \mathbf{A} \rangle$ and there is no $i \in \mathsf{I}$ such that both $\alpha_{\wedge}(i,C_1) \in \langle \mathbf{A} \rangle$ and $\alpha_{\wedge}(i,C_2) \in \langle \mathbf{A} \rangle$

S_7: (negativity) for $C_1,C_2 \in \mathbf{S}$ there is a set of axioms $\alpha_1,\ldots,\alpha_n$ such that $\alpha_{\oplus}(C_1,C_2) \in \langle \mathbf{A} \cup \{\alpha_1,\ldots,\alpha_n\} \rangle$

3.3 Design Decisions

A major design goal for the presented ontology model was to achieve simplicity for use cases in which an overly formal presentation of an ontology would be hindering. This simplicity is employed in a gradual way providing several layers of detail that can be chosen to present the constituents of an ontology at the desired granularity.

At the top-most level (1), an essential distinction is made between the vocabulary elements in the signature **S** and the actual statements that use these elements to express knowledge about the domain as axioms **A**. Hence, this level is appropriate when viewing an ontology as a semantic vocabulary without further distinction of the elements in **S** in terms of concepts, relations, etc., or when working with the statements about the domain in **A** as such without investigating their axiomatic structure or semantic properties. This level already provides sufficient detail for many Semantic Web applications, such as semantically enhanced document retrieval where the elements of an ontology's signature are used for indexing. A particularly crucial design decision at this level is the clear separation of axioms, which are often neglected in use cases associated with light-weight semantics. In our model, axioms are the statements about the domain, whereas entities in the signature form the vocabulary used within these statements, and the two are clearly distinguished.

At the second level (2), the distinction between classes C, instances I and properties P is introduced, which is an essential feature in almost all forms of conceptual knowledge representation ranging from UML in software design to logically expressive ontology languages like OWL. Hence, this level introduces the notions of instantiation and interrelation, and is appropriate when explicitly working with the constituents of an ontology's signature as classes, instances and properties. This level will be sufficient for most Semantic Web applications that exploit the structural features of an ontology but do not rely on the formal semantics that underlies the left out axiomatic details of the elements in **A**. Examples are various works on ontology mapping, Web 2.0 or many approaches that take a graph-oriented view on ontologies as interconnected classes and their instances.

At the third level (3), the participants of instantiation, namely classes and instances, are further split, and abstract concepts $\mathcal{C}$ are distinguished from data types $\mathcal{D}$ like string or integer, while abstract individual objects $\mathcal{I}$ of the domain are separated from concrete data values $\mathcal{V}$. Accordingly, properties are split into relations $\mathcal{R}$, which involve abstract domain objects only, and attributes $\mathcal{T}$, which involve also data values. This design decision reflects the natural separation of abstract domain objects from attributes of such objects, which are merely data values attached to them, and thus accounts for attribution.[1] In this sense, individuals serve as instances of concepts, while data values serve as instances of data types. Hence, this level is appropriate when working with the explicit distinction between abstract objects of the domain and concrete data values, which is often the case in e.g. information integration scenarios where an ontology serves as a unifying schema for databases.

At an even finer-grained level, touching the grounds of formal semantics attached to ontology languages and deduction mechanisms, our ontology model renders the statements about the domain in **A** as expressions of a first-order logical language that employs the elements of an ontology's signature **S** as the symbols for predicates and constants. Although we do not aim at providing a comprehensive ontology language with formal syntax and semantics, we make use of the well established first-order logic notation as a unifying syntactical framework for axioms in an ontology.[2] In this notation the statements of most ontology languages can be expressed and it naturally accounts for the notions of class, relation and instance by means of unary predicates, binary predicates and constants. Furthermore, we provide a minimal set of semantic conditions that formalise the essential characteristics shared by most knowledge representation formalisms and ontology languages in our model, such as the interplay between instantiation and subclassing (S_1,S_2,S_3), the restriction of a relation's domain and range classes (S_4,S_5), the symmetric exclusion of class extensions (S_6) and the ability of deriving class disjointness (S_7).

[1]This distinction is also made in entity-relationship modelling in the field of databases, in object-oriented software design, and in ontology languages like OWL, where object properties are separated from datatype properties.

[2]First-order predicate logic (FOL) has been used as a unifying umbrella formalism for many different forms of knowledge representation, such as description logics or logic programming, which can be understood as syntactic and/or semantic deviations of FOL.

4 CONFORMANCE WITH SEMANTIC WEB RESEARCH

In this section, we demonstrate the compatibility and conformance of our proposed ontology model to existing notions of an ontology in various areas of Semantic Web research, such as ontology language standards, light-weight semantics or ontology mapping. We do so by specifying mappings from different ontology formalisations used in those areas to our simple ontology model.

4.1 Mapping from OWL

The Web Ontology Language (OWL) (Patel-Schneider et al., 2004) is currently the most prominent of the W3C standard languages for expressing ontologies on the web. It provides for expressive knowledge representation based on the underlying *description logic* (DL) formalism (Baader et al., 2003).

In Table 3, we specify how the various elements of an OWL ontology can be mapped to the signature and axioms of our ontology model, where we use the DL style notation for presenting the OWL elements. An OWL ontology is seen as a DL knowledge base *KB* that consists of DL axioms, while $\mathsf{N}_I, \mathsf{N}_C, \mathsf{N}_r, \mathsf{N}_t$ are sets of named entities (individuals, classes, object properties and datatype properties) used within these axioms and D_i are concrete domains that represent datatypes in OWL.

By means of the syntactical first-order logic umbrella that we use for axioms in our model, the DL statements directly map to the axioms in **A**, since DLs are fragments of first-order predicate logic. For named classes $C_{(i)} \in \mathsf{N}_C$, individuals $a_{(i)} \in \mathsf{N}_I$ and properties $r_{(i)} \in \mathsf{N}_r$, we specify some particular DL axioms in terms of the notation introduced in Table 1.

The imposed conditions N_1 - N_4 render OWL as a rather strictly defined ontology language with only binary relations and a clear separation of the different types of entities that prevents any form of metamodelling or reification. Moreover, the conditions S_1 - S_7 ensure all the semantic properties common in rich knowledge representation formalisms, such as transitivity of or propagation of instances along subsumption hierarchies as well as negativity.

Table 3: Mapping OWL.

OWL element	*signature element*
N_C	$\mathcal{C}$
$\{D_1,\ldots,D_n\}$	$\mathcal{D}$
N_I	$\mathcal{I}$
$D_1 \cup \cdots \cup D_n$	$\mathcal{V}$
N_r	$\mathcal{R}$
N_t	$\mathcal{T}$
KB	$\mathbf{A}$
DL expression	*axiom*
$C(a) \in KB$	$\alpha_\wedge(i,C) \in \mathbf{A}$
$r(a_1,a_2) \in KB$	$\alpha_\rightarrow(a_1,r,a_2) \in \mathbf{A}$
$C_1 \sqsubseteq C_2 \in KB$	$\alpha_\triangle(C_1,C_2) \in \mathbf{A}$
$r_1 \sqsubseteq r_2 \in KB$	$\alpha_\triangle(r_1,r_2) \in \mathbf{A}$
$\exists r.\top \sqsubseteq C \in KB$	$\alpha_{D\rightarrow}(r,C) \in \mathbf{A}$
$\top \sqsubseteq \forall r.C \in KB$	$\alpha_{\rightarrow R}(r,C) \in \mathbf{A}$
$C_1 \sqcap C_2 \sqsubseteq \bot \in KB$	$\alpha_\oplus(C_1,C_2) \in \mathbf{A}$
conditions: N_1 - N_4, S_1 - S_7	

4.2 Mapping from RDFS

The Resource Description Framework RDF (Klyne and Carroll, 2004) with its schema extension RDFS (Brickley and Guha, 2004) is the prevalent standard for many applications in the area of linked open data. For an RDF graph G with triples $\langle r_1,r_2,r_3\rangle$, Table 4 specifies the mapping of an RDFS ontology represented by G to the simple ontology model, where R is the set of resources that occur in G and $r_i \in R$ are specific resources.

Table 4: Mapping RDFS.

RDFS element	*signature element*
$\{r_1 \mid \langle r_1, \text{rdf:type}, r_2\rangle \in G \vee$	$\mathcal{I}$
$\quad\langle r_1,r_2,r_3\rangle \in G, r_2 \notin \mathsf{Voc}_{\mathsf{RDF}} \vee$	
$\quad\langle r_3,r_2,r_1\rangle \in G, r_2 \notin \mathsf{Voc}_{\mathsf{RDF}}\}$	
$\{r_1 \mid \langle r_2, \text{rdf:type}, r_1\rangle \in G \vee$	$\mathcal{C}$
$\quad\langle r_1, \text{rdf:type}, \text{rdfs:Class}\rangle \in G \vee$	
$\quad\langle r_1, \text{rdfs:subClassOf}, r_2\rangle \in G \vee$	
$\quad\langle r_2, \text{rdfs:subClassOf}, r_1\rangle \in G \vee$	
$\quad\langle r_2, \text{rdfs:domain}, r_1\rangle \in G \vee$	
$\quad\langle r_2, \text{rdfs:range}, r_1\rangle \in G\}$	
$\{r_2 \mid \langle r_1,r_2,r_3\rangle \in G \vee$	P
$\quad\langle r_1, \text{rdf:type}, \text{rdfs:Property}\rangle \in G \vee$	
$\quad\langle r_1, \text{rdfs:subPropertyOf}, r_2\rangle \in G \vee$	
$\quad\langle r_2, \text{rdfs:subPropertyOf}, r_1\rangle \in G \vee$	
$\quad\langle r_2, \text{rdfs:domain}, r_1\rangle \in G \vee$	
$\quad\langle r_2, \text{rdfs:range}, r_1\rangle \in G\}$	
$\{\text{rdfs:Literal}\}$	$\mathcal{D}$
$<\mathsf{string}>$	$\mathcal{V}$
RDF(S) triple	*axiom*
$\langle r_1, \text{rdf:type}, r_2\rangle \in G$	$\alpha_\wedge(r_1,r_2) \in \mathbf{A}$
$\langle r_1,r_2,r_3\rangle \in G, r_2 \notin \mathsf{Voc}_{\mathsf{RDF}}$	$\alpha_\rightarrow(r_1,r_2,r_3) \in \mathbf{A}$
$\langle r_1, \text{rdfs:subClassOf}, r_2\rangle \in G$	$\alpha_\triangle(r_1,r_2) \in \mathbf{A}$
$\langle r_1, \text{rdfs:subPropertyOf}, r_2\rangle \in G$	$\alpha_\triangle(r_1,r_2) \in \mathbf{A}$
$\langle r_1, \text{rdfs:domain}, r_2\rangle \in G$	$\alpha_{D\rightarrow}(r_1,r_2) \in \mathbf{A}$
$\langle r_1, \text{rdfs:range}, r_2\rangle \in G$	$\alpha_{\rightarrow R}(r_1,r_2) \in \mathbf{A}$
conditions: N_4, S_1 - S_5	

We map the triples to axioms in a way such that plain triples not containing any predefined RDF(S) vocabulary terms are interpreted as assertion axioms connecting instances. Special triples that contain RDF(S) vocabulary like rdf:type, rdf:subClassOf, etc., map to instantiation axioms, subclass axioms, etc.

Of the normative conditions, only N_4 is imposed, which renders RDFS as a rather unconstraint language that allows for reification and metamodelling. The imposed semantic conditions S_1 - S_5 reflect some semantic features of RDFS, such as the instance-class connection along subsumption hierarchies and domain/range properties. Moreover, the lack of conditions S_6,S_7 renders RDFS as a formalism that does not have the expressivity for negation to state or detect logical inconsistencies due to class disjointness.

4.3 Mapping from SKOS

The Simple Knowledge Organization System SKOS (Miles and Bechhofer, 2008) is a prominent light-weight representation formalism in the area of Web 2.0 and light-weight semantics based on RDF(S). In Table 5, we specify a mapping from SKOS concept schemes to our simple ontology model. Similar to the mapping from RDFS, we base the SKOS mapping on an RDF graph G that contains the respective concept scheme, while we take into account specific SKOS vocabulary.

Table 5: Mapping SKOS.

SKOS element	*signature element*
$\{r \mid \langle r, \text{rdf:type}, \text{skos:Concept}\rangle \in G\}$	$\mathcal{I} = \mathcal{C}$
$\{\mathsf{string}\}$	$\mathcal{D}$
$\{\text{skos:related}\}$	$\mathcal{R} = \{r_{rel}\}$
$\{\text{skos:XLabel}, \text{skos:Note}\}$	$\mathcal{T} = \{t_{xLab}, t_{Note}\}$
$<\mathsf{string}>$	$\mathcal{V}$
SKOS RDF triple	*axiom*
$\langle r_1, \text{skos:broader}, r_2\rangle \in G$	$\alpha_\triangle(r_1,r_2) \in \mathbf{A}$
$\langle r_1, \text{skos:narrower}, r_2\rangle \in G$	$\alpha_\triangle(r_2,r_1) \in \mathbf{A}$
$\langle r_1, \text{skos:related}, r_2\rangle \in G$	$\alpha_\rightarrow(r_1,r_{rel},r_2) \in \mathbf{A}$
$\langle r, \text{skos:XLabel}, v\rangle \in G$	$\alpha_\rightarrow(r,t_{xLab},v) \in \mathbf{A}$
$\langle r, \text{skos:note}, v\rangle \in G$	$\alpha_\rightarrow(r,t_{Note},v) \in \mathbf{A}$
conditions: N_1, N_3, N_4, N_5	

Since in SKOS classes are not explicitly distinguished from instances, we let the two respective sets coincide and include all elements from a concept scheme. This allows for using SKOS concepts both in place of classes to build class hierarchies and in place of instances to interlink them. As there are no custom relations or attributes in SKOS, the respective sets comprise the specific SKOS vocabulary only. Subclass axioms reflect the broader/narrower hierarchy essential to SKOS, while other axioms represent relatedness of concepts and their string attributes with the labels, notes, etc. predefined in SKOS.[3]

Besides the imposed normative conditions N_1 for preventing reification, N_3 for separation of data values and their types and N_4 for the restriction to binary predicates, condition N_5 supports the class/instance coincidence mentioned above by preventing concepts from being used to predicate about individuals. On the semantic side, SKOS does not support rich deductive features for its predefined vocabulary, which is reflected by not imposing any semantic conditions.

4.4 Mapping to OA Model

Another area in Semantic Web research is concerned with the mapping and alignment of ontologies (OA), where similar entities across different ontologies are to be identified. We specify a mapping from the formalisation of ontologies in this area, taken from (Euzenat and Shvaiko, 2007), to our simple ontology model. According to this formalisation, an ontology is characterised by classes $\mathtt{C}$, instances $\mathtt{I}$, relations $\mathtt{R}$, datatypes $\mathtt{T}$ and their values $\mathtt{V}$, while specific relation over these sets denoted by $\leq, \perp, \varepsilon, =$ express subsumption, exclusion, instantiation and assertion.

Table 6: Mapping OA Model.

OA model element	*signature element*
$\mathtt{I}$	$\mathcal{I}$
$\mathtt{C}$	$\mathcal{C}$
$\mathtt{R}$	P
$\mathtt{T}$	$\mathcal{D}$
$\mathtt{V}$	$\mathcal{V}$
OM model statement	*axiom*
$(C_1,C_2) \in \leq \cap \mathtt{C} \times \mathtt{C}$	$\alpha_{\triangle}(C_1,C_2) \in \mathbf{A}$
$(p_1,p_2) \in \leq \cap \mathtt{R} \times \mathtt{R}$	$\alpha_{\triangle}(p_1,p_2) \in \mathbf{A}$
$(C_1,C_2) \in \perp \cap \mathtt{C} \times \mathtt{C}$	$\alpha_{\oplus}(C_1,C_2) \in \mathbf{A}$
$(p_1,p_2) \in \perp \cap \mathtt{R} \times \mathtt{R}$	$[\forall x,y : p_1(x,y) \rightarrow \neg p_2(x,y)] \in \mathbf{A}$
$(i,C) \in \varepsilon$	$\alpha_{\wedge}(i,C) \in \mathbf{A}$
$(i_1,r,i_2) \in = \cap \mathtt{I} \times \mathtt{R} \times \mathtt{I}$	$\alpha_{\rightarrow}(i_1,r,i_2) \in \mathbf{A}$
conditions: N_1, N_2, N_4, S_1 - S_7	

The basic elements of the ontology formalisation from the alignment and mapping area are very similar to the signature in our model and map almost one-to-one, while the various axioms are determined by the relations $\leq, \perp, \varepsilon, =$.

Except for the clear separation between object properties and datatype properties, all the normative conditions N_1, N_2, N_4 can be imposed, together with the expressive set of semantic conditions S_1 - S_7 as in the OWL case, which reflects the fact that this formalisation builds on a similar model-theoretic semantics.

[3]In Table 5, skos:XLabel stands for the various label elements and skos:note represents the different documentation properties defined in SKOS.

5 UML VISUALISATION

Next to their formal notation, the presentation of ontologies in research works often also requires the graphical illustration of conceptual models that display an ontology's constituents in an intuitively graspable way, for example, to give an overview on a back-bone taxonomy containing the most essential concepts. The Unified Modelling Language UML[4], well established in the software development community, is a prevalent instrument for displaying conceptual models of all kinds, in particular by means of UML class diagrams, and we propose to use it for this purpose in connection to our formal model.

Applied in the standard way, however, UML class diagrams offer some freedom in the use of their manifold elements, which results in a multitude of different styles of diagrams when used for ontologies, similar to the many different formal ontology models found in research works. This calls for a well-defined uniform way of using the graphical UML elements for displaying at least the essential characteristics of ontologies common to most research contexts. As an additional requirement, the graphical visualisation should be inline with the formal notation and reflect an ontology's formal structure as close as possible, whereas in many presentations ontology diagrams are rather vague with no clear interpretation of graph elements in relation to the underlying formal model.

To provide a uniform way of displaying ontologies in UML, we propose a UML metamodel extension that guides the use of UML class diagrams for graphical presentation of ontologies in accordance to our formal ontology model, as depicted in Figure 2. It defines the entities in **S** of an ontology $\mathcal{O}$ as extensions of predefined UML metaclasses, such as UML::Class or UML::Association in form of new UML stereotypes for classes, instances and properties.[5] The axioms **A** are accounted for by an abstract stereotype with specialisations for the different axiom types $\alpha_{\triangle}$, $\alpha_{\oplus}$, $\alpha_{\wedge}$, $\alpha_{\mathcal{D}\rightarrow}/\alpha_{\rightarrow\mathcal{R}}$ and $\alpha_{\rightarrow}$.[6] The connections between stereotypes for axioms and entities indicate how an axiom is expressed in terms of UML relations. For example, an axiom of type $\alpha_{\wedge}$ is a UML association that involves one instance and one class.

In summary, the UML metamodel extension provides means to visualise the essential characteristics of ontologies in a way consistent to our formal model.

[4]http://www.uml.org/#UML2.0

[5]For simplicity, we left out stereotypes for the more specific concepts, individuals, data values, etc.

[6]The axiom types for domain and range restriction are combined in the graphical model with the idea that a directed association between classes expresses both.

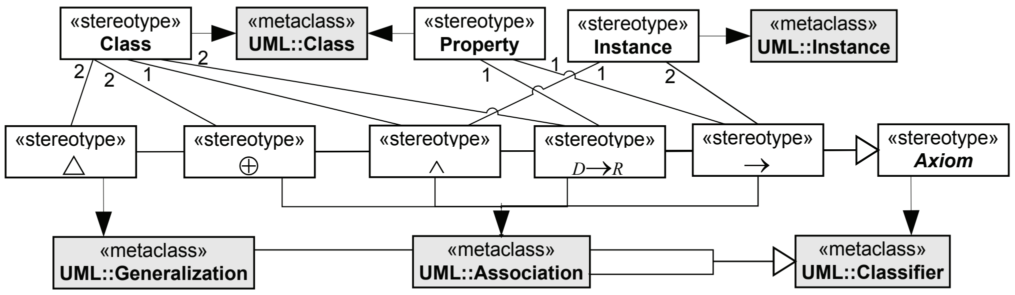

Figure 2: UML metamodel for ontology visualisation.

In this way, we can benefit from the existing tool support for the visual creation of graphical ontology models by means of UML editors, which also enables the possibility for the generation of code in various ontology languages, e.g. through EMF[7] transformations based on the mappings presented in Section 4.

6 AN ONTOLOGY API

Our formal ontology model can also form the basis for a unified programmatic access to ontologies by means of a software API that is layered on top of existing ontology back-end systems. This allows information systems to take a simplified view on ontologies on the software side.

In Figure 3, we sketch an architecture for such an API and its implementation in form of a UML diagram. An ontology and its axioms can be accessed by

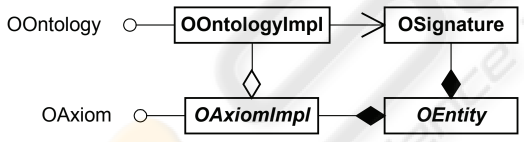

Figure 3: An ontology software API.

respective interfaces whose implementing classes establish the connection to the ontology's signature and entities. For sake of simplicity, we left out specialised types for specific axioms and entities.

Through implementations for various ontology back-end systems, such as e.g. the OWL-API (Horridge et al., 2007), an ontology-based application can access and manipulate ontologies in an easy way, abstracting from the details of the underlying ontology language layer if desired.

[7]http://www.eclipse.org/modeling/emf/

7 CONCLUSIONS

We have proposed a unifying formal model for ontologies that allows for both simplicity in presentation and representation of the essential characteristics of an ontology. We have demonstrated compatibility to SW research works by having provided mappings to various existing ontology languages and formalisations. Moreover, we have sketched the further use of our model as a basis for graphical visualisation in terms of UML and for a simplified programmatic access to ontologies via a specific API.

We argue that our unifying model can help to mediate between different strands of SW research, providing a common view on ontologies that captures their essentials while abstracting from details of specific languages. We aim to work towards a more thorough system of normative and semantic conditions, as well as the realisation of UML-based ontology visualisation and a unifying software API with implementations for various ontology frameworks.

ACKNOWLEDGEMENTS

This work is partially supported by the German Federal Ministry of Economics (BMWi) under the project THESEUS (number 01MQ07019).

REFERENCES

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook*. Cambridge University Press.

Brickley, D. and Guha, R. (2004). RDF Vocabulary Description Language – RDF Schema. http://www.w3.org/TR/rdf-schema/.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer-Verlag, Heidelberg.

Gruber, T. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 6(2):199–221.

Horridge, M., Bechhofer, S., and Noppens, O. (2007). Igniting the OWL 1.1 Touch Paper: The OWL API. In *OWLED*, volume 258 of *CEUR Workshop Proceed.*

Klyne, G. and Carroll, J. (2004). RDF Concepts and Abstract Syntax. http://www.w3.org/TR/rdf-primer/.

Miles, A. and Bechhofer, S. (2008). SKOS Simple Knowledge Organization System Reference. W3C draft.

Motik, B. (2005). On the Properties of Metamodeling in OWL. In *Proceedings of the 4th International Semantic Web Conference (ISWC'05)*, volume 3729 of *LNCS*, pages 548–562. Springer-Verlag.

Patel-Schneider, P., Hayes, P., and Horrocks, I. (2004). OWL Web Ontology Language; Semantics and Abstract Syntax. http://www.w3.org/TR/owl-semantics/.