Extracting Finite Sets of Entailments from OWL Ontologies

Samantha Bail, Bijan Parsia, Ulrike Sattler

The University of Manchester
Oxford Road, Manchester, M13 9PL
{bails,bparsia,sattler@cs.man.ac.uk}

Abstract. The canonical standard description logic reasoning service is classification, that is, the generation of the set of atomic subsumptions which are entailed by some ontology. While this consequence relation is well defined and finite, there is significant variance in the composition of that set. For example, it is common (in tools and in discussion) to exclude some tautologies (e.g., $A \sqsubseteq \top$, $A \sqsubseteq A$). While for many purposes such divergences are harmless, there are many for which precision about what appears in the classification is essential, for example, estimating differences in logical content. In this paper, we propose definitions for different types of finite entailment sets of an OWL ontology based on the transitive closure and transitive reduction of its asserted and inferred class graphs. The purpose of this work is to introduce a flexible and extensible specification for selecting a particular set of entailments, with the aim of ensuring the correctness and replicability of OWL-based applications.

1 Introduction and Motivation

The Web Ontology Language OWL 2 DL is based on the expressive description logic $\mathcal{SROIQ}$ [6]. It is designed to 'facilitate ontology development and sharing via the Web',[1] with OWL development tools aiming at users with little or no knowledge in description logics. *Entailment* is regarded as the 'key inference' of the Semantic Web [12], and while the entailment relation $\models$ is well defined for OWL ontologies [7], misleading nomenclature in ontology tools and anecdotal evidence show that there exist common misconceptions about entailments: first, it is often assumed that the set of entailments of an ontology is finite, and it is possible to extract the set of *all* entailments of an ontology. Second, it is assumed that only non-trivial information is contained in the set of entailments, and tautologies such as $A \sqsubseteq A$ are not entailments. Third, the term entailments is used interchangeably with *inferences*, and the information that is asserted in the ontology is often not considered to be an entailment itself. While the problem of reasoning with and extracting meaningful entailments from inconsistent ontologies has been previously discussed in the literature [8,4], we focus on consistent ontologies for the purpose of this paper, and limit the definitions and examples to atomic subsumptions and equivalences.

[1] http://w3.org/TR/owl2-overview

The ontology editor Protégé 4[2] for instance comes with a 'selected entailments' tab which shows a list of atomic SubClassOf, SubPropertyOf and Type (class assertion) axioms. The tool also offers the option to 'Save inferred axioms as ontology' which saves asserted and inferred axioms as a new OWL ontology. Similarly, Top Braid Composer[3] offers to 'Save [the] inference graph' as a new file. In all cases, it is not clear how the entailments are generated, what the selection criteria is for entailments (inferences), and how the user can modify the ontology to affect these entailments. For example, given an ontology containing only subsumptions and equivalences between named classes, Protégé 4 exports all direct and strict subsumptions between the classes, but offers no options to export the indirect or non-strict subsumptions explicitly. Top Braid Composer, however, does not include (direct or indirect) atomic subsumptions at all in the exported 'inference graph'.

Ontologies that are available on the web may be published as 'compiled' versions, which include the ontology and its entailments of some description. The OWL version of the National Cancer Institute (NCI) Thesaurus, for example, 'includes inferred relationships'.[4] There is, however, no definition of what is regarded as an inferred relationship, how these relationships are determined, and what the selection criteria is. This may leave users wondering what kinds of information they are dealing with, and what implications this has for their understanding of the ontology.

Analytical applications that are based on justifications (minimal subsets of the ontology that are sufficient for the entailment to hold) extract the *entailed atomic subsumptions* of an ontology and compute the justifications therefore [3,5]. Again, transparency of the entailment extraction process is vital for these applications in order to ensure correct and meaningful results. For instance, the number of entailments, as well as the number and properties of their justifications, can be skewed by including or excluding subsumptions caused by unsatisfiable classes, not distinguishing between direct and indirect subsumptions, whether unsatisfiable classes are treated as a subclass or equivalent to bottom, and similar selection criteria. Furthermore, imported ontologies add to the complexity of the problem: computing entailments from the imports closure of an ontology also considers the entailments and justifications of imported ontologies. This may distort the actual number and types of entailments, while also adding a computational overhead to entailment extraction procedures.

The above examples demonstrate how the notion of entailments is widely used in OWL applications, however without a clear understanding of how the entailment relation relates to the set of axioms that is obtained from an ontology. In order to ensure the correctness and replicability of data based on the entailments of an ontology, it is necessary to explicitly specify which (finite) subset of the set of all entailments should be extracted from the ontology. In this paper, we discuss the different aspects of extracting entailments of OWL ontologies and

[2] http://protege.stanford.edu
[3] http://topquadrant.com/products/TB_Composer.html
[4] http://evs.nci.nih.gov/ftp1/NCI_Thesaurus/ReadMe.txt

provide definitions for practical entailment sets. We propose ways of dealing with imported entailments based on the justifications for the entailment. Rather than providing an exhaustive definition for all possible entailment sets of a $\mathcal{SROIQ}$ ontology, the focus of this paper is to encourage clarity when using the term 'entailments' in the context of OWL ontology applications.

2 Applications

2.1 Inferred Ontology Generation in the OWL API

The OWL API[5] provides the convenience class `InferredOntologyGenerator`, which allows users to 'fill' a new ontology with the desired type of entailments, such as inferred atomic SubClass axioms and ClassAssertions. By default, this method only retrieves the *direct* superclasses of a named class when using `InferredSubClassOfAxiomGenerator`. For each class that does not have any direct superclasses other than $\mathsf{OWL{:}Thing}$, the reasoner returns a node labelled $\mathsf{OWL{:}Thing}$. While this method provides a common basis for computing the *inferred ontology*, it does not offer any flexibility for the user to specify which relationships should be included in the output. We propose the implementation of more specific and flexible entailment generation methods in the OWL API as an addition to or extension of existing methods, in order to allow users to conveniently extract clearly defined finite entailment sets from an OWL ontology.

2.2 Presenting Entailments to End-Users

The OWL ontology editor Protégé 4 provides a view of 'selected entailments' of the ontology. As the editor offers no further explanation to how these entailments were extracted in the classification process, this view does not support understanding of the ontology. It may even seem surprising to the end-user that some trivial axioms, such as $\mathcal{A} \sqsubseteq \mathsf{OWL{:}Thing}$, are displayed in the panel while others are missing. A more detailed and modifiable view could support users in exploring the class hierarchy when attempting to understand entailment relations in the ontology.

2.3 Explanation of Entailments

Explanation of entailments for the purpose of debugging a description logic ontology has been the focus of research since the early applications of description logics for modelling domain knowledge [9,10,11]. Most OWL ontology editors provide explanation facilities presenting the part of the ontology which causes the entailment to hold. It may be argued that, from a user perspective, a crucial part of understanding why an entailment holds in an ontology is to have a clear understanding of the notion of entailments, while also being able to control the method of extracting the entailment.

[5] http://owlapi.sourceforge.net

2.4 Metrics

Analytical applications that consider the number and type of entailments in order to infer semantic ontology metrics benefit from clearly defined entailment sets in two ways: first, the basis of the measurements, i.e. *what exactly* is measured, is well defined and transparent, therefore ensuring consistent measurements which are independent from a particular implementation or individual modifications of the results provided by the OWL API. Second, greater flexibility allows to extract entailments that are fit for a specific purpose. For instance, when describing the inferential power of an ontology, it is not necessary to consider the asserted entailments as they hold no information value. On the other hand, to explore the justificatory structure of an OWL ontology, we need to consider that there may be non-obvious and possibly complex reasons for entailments that are asserted in the ontology; this makes it necessary to compute the justifications for both inferred and asserted entailments in order to capture these 'hidden' justifications.

3 Extracting and Counting Entailments

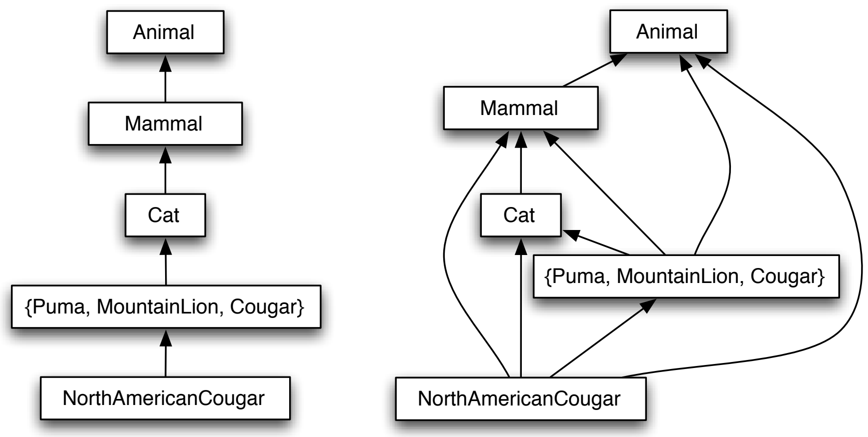
In this section we present different criteria for defining the set of entailments of an OWL ontology. We provide four definitions for finite entailment sets based on the class graph of the ontology, which we then illustrate with examples. In each case, we expect the input to be an OWL 2 DL ontology $\mathcal{O}$, with the output being a set of OWL 2 DL axioms α. Please note that for the purpose of demonstrating our approach, we only focus on *atomic* entailments, i.e. relations between named atomic classes in the ontology.

3.1 Entailments of Description Logic Ontologies

In the remainder of this paper the letters A, B denote class names, C, D (possibly complex) concepts, a an individual, $\mathcal{O} = (\mathcal{T},\mathcal{A})$ a description logic ontology which is the union of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, α an axiom in $\mathcal{O}$, and $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ an interpretation of an ontology $\mathcal{O}$. The notations for $\top$ and OWL:Thing, and $\bot$ and OWL:Nothing are used interchangeably.

The term *finding entailments* of a DL ontology summarises different reasoning tasks, such as the *subsumption problem*, *equivalence* and *satisfiability* checking with respect to a TBox $\mathcal{T}$, and *instance checking* with respect to an ABox $\mathcal{A}$.

Entailment relations in $\mathcal{SROIQ}$ are defined based on the formal semantics given by an interpretation $\mathcal{I}$ [2]. An ontology $\mathcal{O} = (\mathcal{T},\mathcal{A})$ entails that a (possibly complex) concept C is subsumed by a concept D, written as $\mathcal{O} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$. Similarly, $\mathcal{O}$ entails that C is equivalent to D if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$. A concept C is entailed to be *unsatisfiable*, i.e. $\mathcal{O} \models C \equiv \bot$ (commonly expressed as $\mathcal{O} \sqsubseteq C \equiv \bot$) if $C^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{O}$. Regarding instance checking for the ABox $\mathcal{A}$, $\mathcal{O}$ entails that an individual



(a) Asserted Class Graph (b) Inferred Class Graph

Fig. 1. Asserted and Inferred Class Graphs

a is an instance of a concept C ($\mathcal{O} \models C(a)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{O}$. The set of entailments of an ontology is therefore the set of all axioms α such that $\mathcal{O} \models \alpha$.

3.2 Inferred and Asserted Class Graphs

The asserted class graph of an ontology $\mathcal{O}$ is a labelled directed acyclic graph $G = (V, E, L)$ with nodes labelled with (a non-empty set of) class names, including $\top$ and $\bot$, from the signature of $\mathcal{O}$. The graph is initialised by creating a node u for each class name in the signature of $\mathcal{O}$, with the class name being in the label $L(u)$ of the node. An edge (u, v) is added if it is asserted in $\mathcal{O}$ that $A \sqsubseteq B$ for some $A \in L(u)$, $B \in L(v)$, where A and B are class names, $\top$, or $\bot$. For any two nodes u, v in the graph with $L(u) = \{A\}$, $L(v) = \{B\}$, the nodes are collapsed into a single node w if the ontology contains the two subsumption axioms $A \sqsubseteq B$ and $B \sqsubseteq A$, or the equivalence class axiom $A \equiv B$. This leads to existing edges (u, x), (v, y) for some node x, y in the graph, being replaced by the corresponding edges (w, x), (w, y).

The inferred class graph $G' = (V', E', L')$ of the ontology contains an edge (u, v) if $\mathcal{O} \models A \sqsubseteq B$ for some $A \in L(u)$, $B \in L(v)$. A class name A is in the label $L(u)$ of a node u in the inferred class graph if $\mathcal{O} \models A \equiv B_i$ for all B_i in $L(u)$.

The asserted and inferred class graphs in Figure 1 are based on the following toy ontology:

Example 1 (Toy ontology)

$\mathsf{NorthAmericanCougar} \sqsubseteq \mathsf{Cougar}$	$\mathsf{Mammal} \sqsubseteq \mathsf{Animal}$
$\mathsf{Cougar} \equiv \mathsf{MountainLion}$	$\mathsf{Puma} \equiv \mathsf{Cougar}$
$\mathsf{Puma} \sqsubseteq \mathsf{Cat}$	$\mathsf{Cat} \sqsubseteq \mathsf{Mammal}$

3.3 Generating Axioms From the Transitive Reduction and Transitive Closure

The inferred and asserted class graphs of an ontology are uniquely defined. Retrieving these class graphs may be sufficient for counting entailments, as every subsumption relationship between nodes is represented by a single edge, and the arity of a node label (i.e. the number of distinct class names in the node) is equal to the number of equivalent classes. OWL applications however generally present the asserted and inferred class hierarchy as sets of OWL axioms, which need to be generated from the relationships in the respective class graphs.

Generating axioms from the transitive closure of the inferred class graph is straightforward: a SubClassOf axiom is created for each pair of class names in the label of the sub- and superclass node respectively, and an EquivalentClasses axiom for each pair of class names in the label of a node. Example 2 demonstrates how this method can quickly lead to a large set of entailments. In some situations however it is sufficient and more economical to use a subset of these relations based on the transitive reduction of the graph.

The transitive reduction of a directed acyclic graph is a canonical representation for the paths in the graph [1]. The main challenge here is: how can we express, in one single axiom, an edge between nodes which are labelled with multiple class names? Furthermore, if a node is labelled with several equivalent classes, we would like the number of axioms generated from this node to reflect how many equivalent classes the label contains; therefore, multiple nodes that contain different numbers of class names in their labels cannot be represented by a single EquivalentClasses axiom.

For the purpose of expressing subsumptions in the transitive reduction, a function $Rep(u)$ is introduced which selects a single class name from the label of a node u to act as a representative for the node. This function is intended to be user-defined and may retrieve a randomly selected element, the first element in a lexicographical ordering, or even a freshly generated class name (such as the concatenation of the class names in the node), to name a few examples.

In order to generate EquivalentClasses axioms from a node with arity n, where $n > 2$, we apply a function $Pairwise(u)$ to the node label. This introduces an ordering $<$ on the class names in the label of the node (such as a lexicographical order) and returns a set of pairs of class names (A_i, A_{i+1}) where $A_i < A_{i+1}$. While OWL 2 allows EquivalentClasses axioms with an arity greater than two, we choose to express equivalences in binary axioms, which corresponds to the description logic notation of $A_i \equiv A_{i+1}$.

3.4 Top, Bottom, and Tautologies

While an unsatisfiable class is generally referred to as being a *subclass of Bottom*, the class is in fact equivalent to the bottom node OWL:Nothing, as discussed above. We consider this in our definitions and treat an unsatisfiable named class not as a subsumption, but as an atomic equivalence. Likewise, a *universal* class, i.e. a class that is equivalent to OWL:Thing, is not treated as *superclasses of Top*, but as an equivalent class.

Furthermore, tautologies such as $A \sqsubseteq \top$, $\bot \sqsubseteq A$ and $A \sqsubseteq A$ for all named classes A are not included in the entailment set, as they do not hold any information value.

3.5 Different Types of Entailment Sets

We define the set of inferred atomic entailments of an ontology $\mathcal{O}$ as the union of the inferred atomic subsumptions Sub and the inferred atomic equivalences $Equiv$ for the asserted and inferred class graphs $G = (V, E, L)$ and $G' = (V', E', L')$ respectively. The following definitions are ordered by two aspects: whether they are based on the transitive closure $Tc(E')$ or transitive reduction $Tr(E')$ of the inferred class graph, and whether they include (A^+) or exclude (A^-) asserted subsumptions and equivalences respectively.

Transitive closure, inferred, including asserted

$$\begin{aligned}\mathsf{SubTcA}^+(\mathcal{O}) :=&\{A \sqsubseteq B \mid \textit{ there is } (u,v) \in E', A \in L'(u), B \in L'(v),\\ &A \neq B, A \neq \bot, B \neq \top\}\\ \mathsf{EquivTcA}^+(\mathcal{O}) :=&\{A \equiv B \mid \textit{ there is } u \in V', A, B \in L'(u), A \neq B\}\end{aligned}$$

Transitive closure, inferred, not including asserted

$$\begin{aligned}\mathsf{SubTcA}^-(\mathcal{O}) :=&\{A \sqsubseteq B \mid \textit{ there is } (u,v) \in E', A \in L'(u), B \in L'(v)\\ &(u,v) \notin E, A \neq B, A \neq \bot, B \neq \top\}\\ \mathsf{EquivTcA}^-(\mathcal{O}) :=&\{A \equiv B \mid \textit{ there is } u \in V', A, B \in L'(u), A \neq B,\\ &\textit{there is no } v \in V \textit{ s.t. } A, B \in L(v)\}\end{aligned}$$

Transitive reduction, inferred, including asserted

$$\begin{aligned}\mathsf{SubTrA}^+(\mathcal{O}) :=&\{A \sqsubseteq B \mid \textit{ there is } (u,v) \in TR(E'),\\ &A = Rep(u), B = Rep(v), A \neq B, A \neq \bot, B \neq \top\}\\ \mathsf{EquivTrA}^+(\mathcal{O}) :=&\{A \equiv B \mid \textit{ there is } u \in V', (A,B) \in Pairwise(u), A \neq B\}\end{aligned}$$

Transitive reduction, inferred, not including asserted

$$\begin{aligned}\mathsf{SubTrA}^-(\mathcal{O}) :=&\{A \sqsubseteq B \mid \textit{ there is } (u,v) \in TR(E'),\\ &A = \mathit{Rep}(u), B = \mathit{Rep}(v), (u,v) \notin E, A \neq B, A \neq \bot, B \neq \top\}\\ \mathsf{EquivTrA}^-(\mathcal{O}) :=&\{A \equiv B \mid \textit{ there is } u \in V', (A,B) \in \mathit{Pairwise}(u), A \neq B,\\ &\textit{there is no } v \in V \textit{ s.t. } A, B \in L(v)\}\end{aligned}$$

3.6 Examples

The properties of different entailment sets are demonstrated using the above toy ontology as an example.

Example 2 (Transitive closure, including asserted, 18 axioms)

Cougar $\equiv$ MountainLion	Cougar $\equiv$ Puma
MountainLion $\equiv$ Puma	Puma $\sqsubseteq$ Cat
Puma $\sqsubseteq$ Mammal	Puma $\sqsubseteq$ Animal
MountainLion $\sqsubseteq$ Cat	MountainLion $\sqsubseteq$ Mammal
MountainLion $\sqsubseteq$ Animal	Cougar $\sqsubseteq$ Cat
Cougar $\sqsubseteq$ Mammal	Cougar $\sqsubseteq$ Animal
NorthAmericanCougar $\sqsubseteq$ Puma	NorthAmericanCougar $\sqsubseteq$ MountainLion
NorthAmericanCougar $\sqsubseteq$ Cougar	NorthAmericanCougar $\sqsubseteq$ Cat
NorthAmericanCougar $\sqsubseteq$ Mammal	NorthAmericanCougar $\sqsubseteq$ Animal

The transitive closure, including asserted, makes explicit the relationships between every single class in the ontology. It is the largest finite entailment set to be extracted from the class graph. The alternative variant of this set excluding asserted entailments simply discards the axioms that occur in the original ontology, yielding a set of 12 axioms.

Example 3 (Transitive reduction, including asserted, 6 axioms)

NorthAmericanCougar $\sqsubseteq$ Cougar	Cougar $\equiv$ MountainLion
MountainLion $\equiv$ Puma	Puma $\sqsubseteq$ Cat
Cat $\sqsubseteq$ Mammal	Mammal $\sqsubseteq$ Animal

The entailment set based on the transitive reduction of the class graph uses representative elements from each node to produce a minimal representation of the class hierarchy. In this example, the function selects the class names that are asserted to be in SubClassOf relationships in the ontology, otherwise it selects a random class name from the node. The function $\mathit{Pairwise}(u)$ applies a lexicographical ordering on the class names in each node, as described above.

3.7 Counting Entailments

Having translated the class graph into a set of OWL axioms based on the above definitions, the number of entailments can be computed in an unambiguous way. By choosing a representative class name for each node in the transitive reduction, the number of entailed atomic subsumptions is equal to the number of edges in the graph, i.e. one edge in the graph is represented by one axiom.

For the transitive closure of the class graph, the number of entailed subsumption axioms is the sum of all $n(u) * n(v)$ for each edge (u, v) in the graph, with $n(u)$ the number of class names in a node u. The number of entailed binary equivalence class axioms is $n(u) * (n(u) - 1)/2$ for each node u in the graph.

3.8 Implementation

We have implemented the entailment extractor methods using the OWL API.[6] The code is intended to be used with any OWL reasoner that is compatible with the current version of the OWL API. Preliminary tests with large ontologies such as the NCI Thesaurus show that all types of entailment sets can be extracted in practical time.

4 Dealing with Imports

Another issue that needs to be dealt with when extracting and counting entailments from an ontology is its import structure. An OWL ontology $\mathcal{O}$ (the 'root' ontology) that imports another OWL ontology $\mathcal{O}$' can have different kinds of entailments: those that hold in $\mathcal{O} \setminus \mathcal{O}$' (*native* entailments), those that are entirely from the imported ontology, i.e. they hold in $\mathcal{O}$' $\setminus$ $\mathcal{O}$ (*imported* entailments), and those that hold in $\mathcal{O} \cup \mathcal{O}$' but not in $\mathcal{O} \setminus \mathcal{O}$' (*mixed* entailments). When performing analytical tasks on the root ontology such as analysing its inferential power, it may be considered misleading to include the number of imported entailments. Furthermore, if the imported ontology itself imports another ontology (and so on), we will almost certainly obtain data that is not relevant to our original root ontology. While this may not be problematic for some tasks, the origin of entailments needs to be at least made obvious in a way such that the user can make their own judgements on how to handle them.

4.1 Classification of Imported Entailments

We propose a classification of these three types of entailments in an ontology which is based on the notion of justifications for an entailment. A justification is a minimal subset of the ontology that is sufficient for the entailment to hold [10]. The 'origin' of an entailment given an ontology imports structure is determined by the set of its justifications.

[6] The source code is available for download and modification at http://code.google.com/p/owl-entailment-extractor.

Type 1: Native entailments We may want to restrict the entailment extraction to the root ontology and discard all entailments that originate partly or entirely from the imported ontologies. In this case, the class graph construction and reasoning process is limited to the root ontology axioms only. Computing justifications is not necessary for this type of entailments.

Type 2: Imported entailments While the entailments that originate purely from the imported ontology may not be relevant to application, an analysis of the type and numbers provides information about the computational overhead they may cause when not excluding them from the entailment set. Type 2 entailments are extracted by computing the inferred class graph for axioms that are contained in the imported ontology only. Computing justifications is not necessary for this type of entailments.

Type 3: Mixed entailments In this type, we gather all entailments that are considered 'mixed' for at least one of the following reasons: first, an entailment that has at least one justification which contains axioms from both $\mathcal{O}$ and $\mathcal{O}$' is considered mixed. Second, an entailment that has some justification that comprises axioms from $\mathcal{O}$, and some justification that comprises axioms from $\mathcal{O}$'.[7] Type 3 entailments are computed by extracting all entailments from the union of the imports closure of the root ontology, then sequentially generating justifications for these entailments. If the set of justifications contains axioms from both the root and the import ontologies, the entailment is marked as 'mixed' and no further justifications need to be found.

5 Conclusions and Future Work

Due to the ambiguous use of the term 'entailments' in the OWL community, it is necessary to explicitly specify the selection criteria for entailments in both analytical and user-oriented applications. The methods for extracting entailments from OWL ontologies currently provided by the OWL API and ontology development tools provide little flexibility and do not support understanding of the entailment relationships in the ontology. We have presented well-founded and extensible definitions for different types of entailment sets of an OWL ontology, based on its class graph. Depending on the purpose, users can extract entailments from the transitive reduction or the transitive closure of the ontology, and decide whether the asserted entailments should be included. We have also introduced different ways of dealing with entailments that are partly or entirely caused by imported ontologies. The proposed methods offer flexibility and transparency when handling entailments, which may support ontology understanding as well as clarify analytical tasks.

[7] While this distinction is not relevant to the classification of entailments, it does matter in the context of analysing the structure of justifications in an ontology.

Thus far, we have discussed definitions and examples for entailed subsumptions between atomic classes. In order to capture the wide range of entailments from an expressive description logic such as $\mathcal{SROIQ}$ and to provide extensive information about an ontology, these definitions can be extended in two directions: first, to cover the expressivity of OWL 2 DL ontologies beyond subsumptions between named classes, such as class assertions, object property hierarchies and data property hierarchies. Second, we may also want to capture non-atomic entailments, such as literals (disjointness of classes) and subsumptions and equivalences between complex class expressions (e.g. existential and universal restrictions on atomic class names). In the case of complex class expressions, it will be necessary to identify which complex entailments are of interest to users, depending on the needs of a particular application.

References

1. A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.
2. F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi. *The description logic handbook: theory, implementation, and applications.* Cambridge University Press, 2003.
3. S. Bail, B. Parsia, and U. Sattler. The justificatory structure of OWL ontologies. In *Proc. of OWLED-10*, 2010.
4. P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure. A framework for handling inconsistency in changing ontologies. volume 3729 of *Lecture Notes in Computer Science*, pages 353–367. Springer, 2005.
5. M. Horridge, B. Parsia, and U. Sattler. The state of bio-ontologies. In *To be published in Proc. of ISMB-11*, 2011.
6. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In *Proc. of KR-06*, pages 57–67, 2006.
7. I. Horrocks and P. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. of Web Semantics*, 1(4):345–357, 2004.
8. Z. Huang, F. Van Harmelen, and A. Teije. Reasoning with inconsistent ontologies. In *Proc. of IJCAI-05*, volume 19, page 454. Citeseer, 2005.
9. D. McGuinness and A. Borgida. Explaining subsumption in description logics. In *Proc. of IJCAI-95*, volume 14, pages 816–821, 1995.
10. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proc. of WWW-05*, pages 633–640, 2005.
11. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI-03*, pages 355–362, 2003.
12. E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. of Web Semantics*, 5(2):51–53, 2007.