Ontology coverage check: support for evaluation in ontology engineering

Viktoria PAMMER, Peter SCHEIR, Stefanie LINDSTAEDT
Know-Center, Infeldgasse 21A, 8010 Graz, Austria
E-mail: {vpammer, pscheir, slind}@know-center.at

Abstract.
Support for the process of ontology engineering is needed in order to reduce the effort still necessary to build an ontology. Some can be given by facilitated evaluation of the ontology under development. To that purpose we present an automated method that supports data driven ontology evaluation by checking to what extent the concepts and axioms of the ontology under evaluation are covered by a given set of individuals (data).

We applied the here presented ontology coverage check (OCC) to various ontologies and will report on the results. The results highlight not only the potential of OCC but also some characteristics of ontologies currently available to the public.

Introduction
Ontology engineering has received recent attention through the semantic web, although knowledge engineering as such has a long tradition in the scientific field of artificial intelligence. Sound formalisms for knowledge representations were established and the term “ontology” started being used for knowledge models. Although many slightly differing definitions of what an ontology is exist, it is most commonly understood to be “a formal, explicit specification of a shared conceptualization” [15].

Numerous applications for ontologies are conceivable. The envisaged advantages make ontologies interesting not only for academia but also for industry. Gruber [5] sees the advantage of ontologies in sharing and reusing of knowledge. Uschold and Gruninger [16] consider as main benefits of ontologies the possibility of enhanced support for communication between humans, the interoperability between systems, the reuse of existing schemes, the automated consistency checking and the support in specifying an IT system. Noy and McGuinness [12] see similar advantages, like sharing a common understanding of the structure of information among people or software agents, enabling reuse of domain knowledge, making domain assumptions explicit, separating domain knowledge from the operational knowledge and easily analyzing domain knowledge. These classic arguments for ontologies also hold for non-web environments, as is the case in our application context of work-integrated learning.

Despite the huge interest in formal knowledge representation, ontologies still lack industrial application. This is in our opinion directly related to the effort needed to create an appropriate ontology. Porzel and Malaka [13] identify two critical issues that restrain the widespread application of ontologies: First the creation of an ontology in terms of the well known knowledge-acquisition bottleneck (see Edwards [3]). Second the evaluation of given ontologies regarding a specific purpose. Recent research addressing the first issue has lead to the creation of a whole new discipline known as ontology learning, which deals with the
(semi-)automatic creation of domain models from, mostly, text [10].

We offer support for the first of these issues (knowledge bottleneck) by using the second of these issues (evaluation). Our goal is to provide semi-automated support for the process of ontology engineering. Our vision is a holistic, semi-automated ontology engineering tool, as in our minds ontology evaluation is an integral part of the ontology engineering process. In the work presented here we try to contribute to facilitating ontology engineering by evaluation of the ontology under development. To that purpose we have developed the ontology coverage check (OCC), which is an automated method for evaluating the fitness of the ontological schema (concepts, axioms) to a given set of individuals.

Currently available support for ontology engineering mainly consists in standard vocabularies like the W3C recommendations RDF, RDF Schema and OWL. A number of ontology modeling tools such as OntoStudio [1], Protégé [2] and SWOOP [3] support the ontology engineer in creating syntactically correct ontologies and there are reasoners like Racer [4], FaCT++ [5] or Pellet [6] to check for logical consistency and satisfiability. On top of that, we contribute the ontology coverage check (OCC) which is an automated method that helps checking logical restrictions for their meaningfulness.

The rest of the paper is structured as follows: First we will give an overview over related work on ontology evaluation and present the use case for our approach to ontology evaluation in Chapter 1. Then we will describe the OCC method in more detail and illustrate it by an example in Chapter 2. We will report on the results of applying OCC to various ontologies in Chapter 3. Relevant findings and future work will be discussed in Chapter 4. Finally we will review the main issues and conclude in Chapter 5.

1 Motivation for OCC in context of general ontology engineering in general and a specific ontology-driven application

1.1 Related work

Ontology evaluation is necessary in a variety of situations: during the process of ontology engineering to decide whether the product is satisfactory, at the point of using an ontology in order to be sure to use an appropriate one or at the point of developing an automated ontology learning method whose outcome needs to be measured [1].

There are various ways of differentiating between ontology evaluation methods: according to which user they address, at which stage in the lifecycle of an ontology the method is to be applied and which properties of the ontology it evaluates [4]. Brank et al. differentiate between approaches depending on what the ontology is evaluated against: a human expert, a golden standard, an application or data [1]. Staab et al. describe a broad spectrum of aspects, targeting different audiences: ontology engineers, domain experts or ontology consumers [14].

A very formal method is OntoClean, first presented by Guarino and Welty in 2000 [6]. It is an evaluation method that clearly targets ontology engineers, and is useful for validat-
ing the taxonomic relations within an ontology. OntoClean involves annotating all concepts of an ontology with formal properties such as essence, rigidity, identity and unity. A such-annotated ontology can be automatically checked for correct taxonomic relations according to a given set of rules. Its main drawback, tedious manual annotation of all concepts of the ontology, has recently been alleviated by Völker et al. [17] who have devised a method to automatically annotate the concepts as required by OntoClean.

Among the existing approaches to evaluation, we can most closely relate our work to the approach of data driven ontology evaluation described by Brewster [2]. They describe an approach to choose among ontologies according to their fitness for a given text corpus. We share two common points with Brewster. First the idea that an ontology should match a very concrete domain defined by a text corpus. Second, that the extent to which the data cover the ontology is a measure of appropriateness of the ontology.

However, there are various differences. Some of them are due to the fact that Brewster’s method is designed for ontology selection instead of for ontology engineering. Another difference is, that the OCC does not use data outside the ontology (like a text corpus for example), but assumes the data to be already available in an ontology format, i.e. as individuals in OWL terminology. On the other hand, the OCC is able to consider axiom usage by the data, which so far is not considered by any other evaluation method we know of.

1.2 Use case for document based ontology engineering and need for evaluation

We introduce Eve as actor of our use case. She is part of a community in which a lot of shared documents exist. Currently, this document base is searched by keywords or according to folder structure. Eve wants to facilitate the usage of these documents for the community by transforming the implicit knowledge stored in documents to explicit knowledge (ontology). Relevant documents can then be found not only by keyword but by topic (ontology concept). As added value, community members can navigate through the documents following the relations defined in the ontology.

Eve is rather an “amateur” ontology engineer than an expert. She is not specifically trained in ontology engineering. Although Eve is of course knowledgeable in the domain of interest, she is not a domain expert either. Eve takes interest in creating an appropriate ontology merely because she has an interest in creating a learning/working environment in which relevant material can easily be found.

Note that the community sharing a document base can be formal like a company or more informal like a community of practice.

Creation of a domain ontology consists basically of formalizing knowledge implicit in a domain. Eve starts by identifying relevant concepts, relations, individuals and facts (assertions). She moves on to formally represent this knowledge in OWL DL.

The knowledge implicit in the shared documents is available for incorporation into the domain ontology. Using it will ensure that the ontology captures a shared view on the domain of interest. Then, Eve needs feedback on the quality of the created domain ontology. This quality measure should express how well the ontology represents the knowledge implicitly available in the shared document base.

The goal of our work is to develop automated methods to support Eve’s work. As seen above, the ontology engineer we have in mind is not necessarily an expert ontology engineer.
So, feedback is definitely needed on how well the formal ontology represents the knowledge implicitly available in the shared document base.

The requirements for a desirable ontology evaluation method according to the use case stated above are therefore:

1. Special regard needs to be put to the fact that the ontology engineer is not an expert in ontology engineering.

2. The evaluation method shall deliver hints on how/where to improve the ontology.

The use case is resumed in Chapter 2.3 where we will describe the contribution of the ontology coverage check to ontology evaluation and specifically to capturing the fitting of the schema to the data.

2 Ontology evaluation using OCC

The ontology coverage check’s intended usage is facilitating evaluation by providing feedback on the percentage of the ontology that is used by individuals inserted for test purposes. It is suggested that a created ontology is populated either manually, semi- or fully-automatically with real-world individuals from the domain of knowledge in order to check for missing concepts, relations and restrictions. We will say that these individuals test the ontology.

We stress here that this is a necessary precondition for a useful application of OCC: the ontology must be populated by meaningful individuals. Meaningful is here defined with regard to the target domain of knowledge of the ontology under development.

The idea of individuals testing the ontology and coverage check originates from the widespread notion of unit testing and test coverage in software engineering. Unit testing is included in an IEEE Standard and has as goal to test each component separately. In test-driven development practices, the paradigm is to cover as much of source code as possible by tests. Source code is covered if it is executed while the test is being run. As an example, for an if-statement to be covered, two test cases are needed, one for each of the two branches (the if-condition succeeds or fails). The test coverage is the percentage of code that is covered in this way. Supporting frameworks make it easy to implement unit tests and to check the source code coverage are widely available on the Web.

For ontologies, we have translated source code to concepts, relations and axioms of the ontology which need to be covered by individuals, serving as tests. We draw the analogy that if an individual is of the type of some concept or satisfies a defining axiom, this corresponds to testing the concept/axiom with this individual. The test is successful if the testing individual is a meaningful, real-world individual, i.e. if it is not specifically engineered to fit the concept or axiom but rather taken directly from the domain of interest. The ontology coverage check (OCC) therefore checks the percentage to which the ontology is being covered or tested by a given set of individuals. Of course, the OCC is able to output the names of the covered and uncovered concepts and axioms.

---

2.1 Technical description of the OCC

The OCC currently supports two features. First, checking the coverage of concepts (basic check) which determines how many and which concepts have individuals. Second, checking the usefulness of axioms (extended check), which determines defining axioms that can be used for inferring the type of an individual for at least one of the given individuals. Relations are not considered at the moment, but this is definitely a to-do in the ongoing work.

Basic OCC: Concept coverage

A concept is covered if it either has direct individuals or its subclasses have direct individuals of its type. The feedback this check can give to the ontology engineer is, which parts of the ontology are already populated, and which not. This is particularly useful if the population is done in a semi- or fully automated way.

The OCC will output the percentage of covered and uncovered concepts and, if desired, the names of covered and uncovered concepts.

Extended OCC: Axiom usage

The axiom usage check relies on what we call a defining axiom. A defining axiom is a necessary and sufficient condition for an individual to be of a certain type. In OWL vocabulary, this means that the owl:Restriction is an equivalent class to the concept which constitutes the type for the individual.

An axiom is used by the population of the ontology if at least one individual exists whose type can be inferred because of the relations attached to this individual. It does not matter if the individual is actually asserted to be of this type or if the inference really needs to be done by a reasoner. It is sufficient if it is possible to infer the type of the individual because of its relations satisfying the defining axiom.

The OCC will determine the percentage of used and unused axioms and if desired the names of used and unused axioms.

By definition, the defining axioms taken into account by the OCC can only be an existential (owl:someValuesFrom), a quality (owl:hasValue) or a minimal cardinality (owl:minCardinality) restriction. This is derived from the fact that OWL uses the Open World Assumption. A universal restriction (owl:allValuesFrom) can never be known to be true, because a relation attached to the individual in the future or a yet unknown relation could always change the inference. However, inferences may not change by adding new facts to the ontology. Universal restrictions as well as cardinality (owl:cardinality) or maximum cardinality (owl:maxCardinality) restrictions are used in reasoning to detect the inconsistency of an individual. Therefore, as it is not possible to infer the type of an individual because of a universal, cardinality or maximum cardinality restriction, these kinds of restrictions are not taken into account by the OCC.

2.2 Illustration of OCC

For clarification, the principles of the implemented ontology coverage check are illustrated in a simple example. Following this, the envisaged usage of the OCC during ontology engineering is illustrated as well.

The example ontology (classes, relations and axioms) is depicted in Fig. 1. It mainly consists of Mammals, which are defined, via necessary and sufficient conditions, by their

[^6]: http://owl.man.ac.uk/2003/why/latest/#6
way of moving. Thus a Person is defined by walking upright, a Dog by walking on four legs and a Whale by swimming (the latter axiom is not depicted). We are of course aware, that this ontology is far from capturing biological realities.

Illustration of basic OCC

In the basic ontology coverage check, only Whale is uncovered, as there is not a single individual of type Whale present in the ontology.

Illustration of extended OCC

In the extended check, restrictions that define concepts are seen as covered if through the use of the defining axiom at least one individual can be inferred to be member of the defined concept. This means that if an individual is asserted to belong to a defined concept, this is not enough for covering the defining axiom. The individual must also be asserted the properties that satisfy the axiom. Consequently, moves hasValue upright, defining Person, is covered as it serves for identifying Fred as a Person. Fred is asserted to be a Mammal that walks upright. According to the axiom moves hasValue upright that defines the concept Person, Fred can be inferred to be a Person. The axiom moves hasValue onFourLegs, defining Dog, is not covered, as it does not fit any individual in the asserted model. Bello is asserted to be a Dog, and it can be inferred therefore that he moves on four legs. But the assertions available on Bello do not satisfy the axiom moves hasValue onFourLegs.

2.3 Illustration of OCC integration in the ontology engineering process

Resuming the use case started in Chapter 1.2 we find Eve at the point of wanting feedback on the created ontology.

The example ontology at this stage consists of schema (Person, Mammal, moves, Person moves upright, Dog moves on four legs, etc.) and data (Fred, Bello, etc.) taken from the real-
world, i.e. extracted from the shared documents that form the informal, shared knowledge of the community. Notice, that the population is done prior to applying OCC.

Eve then applies the OCC to the ontology and detects that Whale is not covered at all. In order to have all concepts covered, an individual of type Whale needs to be inserted. After Eve has done so, she applies the OCC again. This time OCC feeds back that all concepts are covered. Eve can now be sure to have tested every concept. Another possibility is that she notices that in the domain of concern, no individuals of type Whale occur at all, and thus the concept of Whale can be removed from the ontology. This decision is still left to Eve to take.

Furthermore, OCC has detected that the axiom moves hasValue onFourLegs defining Dog is not used. In order to have this axiom covered, Eve can insert more information on Bello (in addition to the plain assertion that it is a Dog). This has the added advantage of immediately noticing if the available relations fit the real dog. Following this, Eve checks again if there are axioms on Dog that are uncovered. For example, the concept Dog could also have been (wrongly) defined by moving on five legs. Eve would realize, that it is impossible to find a real-world individual that can satisfy this axiom defining Dog.

3 Results

The ontology coverage check (OCC) method has been implemented in Java using the Jena framework and the Pellet OWL Reasoner. The Jena framework allows representing the ontology as Java object at runtime. Pellet is used to provide inferences on the ontology. At the moment OCC is without a graphical user interface.

The OCC method has first been applied to one small- and one medium-size ontology built at the Know-Center. Then, the OCC has been applied to two of the four ontologies made available at EON2006 and finally OCC has been applied to two ontologies found on the web.

Numbers of concepts and axioms will be given only as approximate numbers, as the counting of both depends on whether anonymous classes, intersections, owl:Thing, necessary conditions etc. are counted.

3.1 Small- and medium-size ontologies, built at the Know-Center

The OCC has first been applied to a small ontology, built at the Know-Center, which describes the System Goal modeling stream of the RESCUE process. This ontology intended to be used in an ontology based search engine in the project APOSIDLE. The ontology consists of around 15 classes, 3 axioms and 35 individuals 69% of concepts are covered, and 66,6% of axioms are used.

Secondly, OCC was also applied to a larger ontology (of which the previous one is actually an excerpt) which covers the whole of RESCUE. It consists of around 60 concepts, 20 axioms and 70 individuals. 33% of concepts are covered and 10% of axioms are used.

In both cases, the ontologies were still under development and we were using real-world examples like use case descriptions, requirement specifications in order to check if the ontological schema contained exactly what we needed. In both cases, the uncovered concepts

(always excepting owl:Nothing, which is defined to have no individuals\(^\text{13}\) corresponded to those that had not yet been looked at in detail but are still serving as stub concepts.

**Analysis of OCC results**

In the first ontology, there was one unused axiom, and by trying to have it covered, the axiom was found to be not satisfiable by meaningful individuals taken from the real domain.

Although the axiom itself was logically sound and intuitively sensible, it was practically incorrect for the domain of interest. Consequently, the axiom was removed. The domain knowledge that should have been represented by the axiom in question was finally expressed in a different way.

### 3.2 Two medium-size ontologies used at the EON2006

At EON2006, four ontologies have been provided which were to be evaluated by submitting authors. The OCC was applied to two of them; the first one being the rove-Ontology\(^\text{14}\) and the second one being an organizational view on the AIFB\(^\text{15}\).

The rove-Ontology was developed during the 3rd Summer School for the Semantic Web at Cercedilla, and is intended to model the summer school and its participants. It consists of approximately 20 concepts, 10 axioms and 140 individuals. 72\% of classes are covered and 20\% of the axioms are used. Note that nearly all uncovered axioms were maximum cardinality and universal restrictions which, as explained above in section 2.2, can by definition not be considered by the OCC:

The viewAIFB-Ontology provides a metadata vocabulary (schema) about the Institute of Applied Informatics and Formal Description Methods (AIFB) Karlsruhe as well as individuals (data). It consists of around 55 concepts, 68 axioms and 1150 individuals. 51\% of concepts are covered, and no axiom is used.

Both ontologies are in use, and we suppose that they are populated not only with test data but with all individuals currently of relevance. In this case, the OCC is applied for assessment purposes during the lifecycle phase of usage (instead of during development)

**Analysis of OCC results**

Although the second ontology contains quite a lot of axioms, they are unused. This may be explained by the fact that according to the comment at EON2006’s homepage various people use the ontology who are not necessarily familiar with either ontologies or with the available reasoning power.

Nevertheless, this can lead to the question if the power of the system is explored enough or if the ontology could not be simplified, which in turn would improve its usability (for usability in relation with ontologies and evaluation see e.g. [4]).

\(^{13}\) [http://www.w3.org/TR/owl-ref/#Nothing-def](http://www.w3.org/TR/owl-ref/#Nothing-def)

\(^{14}\) [http://www.aifb.uni-karlsruhe.de/WBS/dvr/rove/](http://www.aifb.uni-karlsruhe.de/WBS/dvr/rove/)

\(^{15}\) [http://www.aifb.uni-karlsruhe.de/viewAIFB_OWL.owl](http://www.aifb.uni-karlsruhe.de/viewAIFB_OWL.owl)
3.3 Oyster, implementation of OMV, and SUMO Mid-Level-Ontology

Oyster implements a proposal for a metadata standard, the so called Ontology Metadata Vocabulary (OMV\textsuperscript{16}). OMV has been developed in the EU IST thematic network of excellence Knowledge Web as a vocabulary to describe ontologies. It consists of around 15 concepts, 160 axioms and some 40 individuals. 47\% of concepts are covered and none of the axioms are.

SUMO (Suggested Upper Merged Ontology), see \textsuperscript{11}, is a proposal by a dedicated working group for a generic upper-level ontology\textsuperscript{17}. The SUMO Mid-Level-Ontology is intended to bridge the abstract level given by SUMO and more detailed domain ontologies. It consists of more than 1800 concepts, no axioms and around 300 individuals. 5\% of concepts are covered. We also searched domain ontologies building on SUMO, but as no restrictions are given by the upper-level ontology, low-level ontologies tend not to use restrictions either.

Analysis of OCC results

In this case, direct application of OCC had near to no benefit. Both ontologies, Oyster and SUMO mid-level ontology, are well-designed ontologies which seem to be generally accepted as sensible representations. They are neither under development (where we see the most benefit in applying OCC) nor used within an application which generates individuals. Therefore, as they do not fulfill the necessary precondition of being populated with meaningful individuals, OCC is not a suitable method for evaluating them.

In order to apply OCC on Oyster or SUMO, these ontologies would first need to be populated by meaningful individuals.

We also state here an additional precondition for a useful application of OCC. If the ontology under development reuses an already existing ontology through import, it might happen that most parts of the ontology under development are imported and only a small fraction of the imported concepts is used at all. OCC would constantly output low coverage, which would not be relevant to the ontology engineer however, as most of the concepts/axioms are not even intended to be used. For this case, OCC would at least need to be modified in order to provide useful feedback, as currently imported ontologies receive no special treatment. We base this statement on the observation with some low-level ontologies using the SUMO ontology, which contained large unpopulated areas.

4 Discussion and outlook

First, we note here that the evaluation itself proved to be an unexpected challenge. Most available ontologies are already finished and published as ”schemas” i.e. do not contain meaningful individuals. A lot of them are also upper- or mid-level ontologies which do not contain a representative set of individuals per definition. In this case, the ontology under evaluation would first need to be populated before OCC could be applied.

We see this as one of the, although not wholly intended, outcomes of this work. Our interpretation is, that individuals are seen as (often sensitive) data that one does not publish. Supporting evidence is given by Tim Finin who states that

\textsuperscript{16}http://ontoware.org/projects/omv/
\textsuperscript{17}http://www.ontologyportal.org/
"...the vast majority of defined classes have no immediate instances and the majority of properties have never been used to assert a value." 18

Concerning future work, we think that a study of the characteristics of available ontologies and their actual application is needed. This should provide the basis for developing better-tailored ontology engineering support.

Further issues for future research arise, if we measure the here presented OCC method with regard to the requirements stated in Chapter 1.2.

The first requirement was to consider a non-expert ontology engineer as target user. We offer support for a part of ontology engineering that is not intuitive to many users, namely axioms. In order to offer intuitive support, the first step will be to spend some efforts on developing an easy-to-use graphical user interface (GUI). We intend to integrate OCC as plugin to Protégé. In this case OCC’s results could be illustrated as depicted in Fig. 1. The layout of Fig. 1 is inspired by the actual Protégé plugin Jambalaya. 19 After that, a user study will have to be carried out.

The second requirement was to deliver hints to the user on where to improve the ontology. Currently we discover where to test the ontology. This is not yet an indication that the ontology needs to improved in this place, but an indication that the ontology needs to be tested in this place. Hence, we are halfway to meeting this requirement.

Technically, open issues are to consider relations in addition to concepts and axioms and to modify the OCC in order to enable to make meaningful statements about the usage of universal restrictions, cardinality and maximum cardinality restrictions in the ontology.

5 Conclusion

We presented an automated method for checking the usage of an ontology’s schema entities (concepts, axioms) by a set of individuals. The novel aspect of the ontology coverage check lies very much in extending the meaning of unpopulated areas from concepts without individuals to axioms without satisfying individuals.

By presenting related work as well as giving a specific illustrative use case, we motivated the importance of evaluation as an integral part during ontology engineering. The presented ontology coverage check method addresses ontology engineers and provides feedback while testing the ontology for applicability to the intended domain of knowledge.

We applied the developed OCC method to several ontologies and reported on the results. During application of OCC to ontologies publicly available on the Web, we found evidence that most of the semantic data available are ontology schemata (concepts, relations, axioms) and rarely the corresponding data.

Finally we reach the conclusion that the OCC, while not being a generally applicable ontology evaluation method, does indeed fulfill the purpose for which it was designed. Namely, it supports ontology evaluation in the course of ontology engineering by providing feedback on the extent to which the ontology covers a given set of (test) individuals. Generally speaking, OCC checks the fitting of the ontology schema to given data in an automated way.

18 http://ebiquity.umbc.edu/blogger/2006/08/12/is-there-real-world-rdf-sowl-instance-data/
Acknowledgments

This work has been partially funded by the EU Integrated Project APOSDLE\textsuperscript{20} under grant 027023 in the IST work program.

The Know-Center is funded by the Austrian Competence Center program Kplus under the auspices of the Austrian Ministry of Transport, Innovation and Technology (\url{www.ffg.at/index.php?cid=95}) and by the State of Styria.

References


\textsuperscript{20}Advanced Process Oriented and Self-Directed Learning Environment, \url{http://www.aposdle.org}


