Applied Ontology 12 (2017) 275–298
DOI 10.3233/AO-170190
IOS Press

# Ontohub: A semantic repository engine for heterogeneous ontologies

Mihai Codescu[a], Eugen Kuksa[b], Oliver Kutz[a], Till Mossakowski[b,*] and Fabian Neuhaus[b]
[a] *Free University of Bozen-Bolzano, Italy*
*E-mails: Mihai.Codescu@unibz.it, Oliver.Kutz@unibz.it*
[b] *Otto-von-Guericke University of Magdeburg, Germany*
*E-mails: kuksa@iks.cs.ovgu.de, till@iks.cs.ovgu.de, fneuhaus@iks.cs.ovgu.de*

**Abstract.** Ontohub is a repository engine for managing distributed heterogeneous ontologies. The distributed nature enables communities to share and exchange their contributions easily. The heterogeneous nature makes it possible to integrate ontologies written in various ontology languages. Ontohub supports a wide range of formal logical and ontology languages, as well as various structuring and modularity constructs and inter-theory (concept) mappings, building on the Object Management Group's Distributed Ontology, Model and Specification Language (DOL). Ontohub repositories are organised as Git repositories, thus inheriting all features of this popular version control system. Moreover, Ontohub is the first repository engine meeting a substantial amount of the requirements formulated in the context of the Open Ontology Repository (OOR) initiative, including an API for federation as well as support for logical inference and axiom selection.

Keywords: Ontology repository engine, Git, Linked Data, logical inference, heterogeneity

Accepted by: Mike Bennett

## 1. Introduction

Ontologies play a central role for enriching data with a conceptual semantics and hence form an important backbone of the Semantic Web. The number of ontologies that are being built or are already in use is steadily growing. This means that capabilities for organizing ontologies into repositories, as well as searching, maintaining, structuring, linking and modularizing are becoming more important.

However, for ontology developers there are only limited options to store and maintain their ontologies. Bioportal (Noy et al., 2009) is one of the most successful ontology repositories available. However, it is limited to ontologies for the life sciences that are written in the Web Ontology Language (OWL; OWL Working Group, 2009), Resource Description Framework (RDF; Manola and Miller, 2004), or Open Biological Ontologies (OBO; Smith et al., 2007) format; and it provides only limited support for version control. For this reason, many ontologies are maintained within repositories (e.g., the Common Logic Ontology Repository, COLORE[1]) that are hosted by general purpose version control repository engines such as Github (http://github.com) or Bitbucket (https://bitbucket.org). However, these kinds of repositories treat ontologies just as any other text document and provide no ontology-specific services.[2]

[*]Corresponding author. E-mail: till@iks.cs.ovgu.de.
[1]See http://colore.oor.net.
[2]For a more detailed discussion of Bioportal, the use of Github for hosting ontologies, and other related work, see Section 2.5.

The main idea behind Ontohub is to offer the ontology community a place where ontologies on any subject and written in a variety of formal ontology languages may be published and maintained, and which offers services that make it easier to develop, reason with and test ontologies. In a nutshell, Ontohub is a repository engine that is developed with the needs of the ontology community in mind. Thus, Ontohub provides services similar to Github in that it offers users a distributed version control system for managing their ontologies. Users have complete control over their (public and private) repositories, and Ontohub allows other users to find and reuse published ontologies. Yet, in addition, it is able to parse all major ontology languages and provides reasoning capabilities for many of them. Last, but not least, it supports the reuse and structuring of ontologies via the Distributed Ontology, Model and Specification Language (DOL). This includes important features such as the translation between ontology languages, ontology alignment, and the encoding of competency questions.

The Ontohub architecture is open and flexible because it is decentralised into several communicating RESTful Semantic Web services. Moreover, Ontohub follows linked open data principles (see Section 2.3). Thus ontological and Semantic Web principles have been employed during the design of Ontohub itself. Surprisingly, this is not standard for ontology repository engines.

The main purpose of this paper is to provide an overview of Ontohub.[3] We start with a discussion of the motivation for its design, a general overview of its features and a comparison to similar tools (Section 2). As we already mentioned above, many advanced features of Ontohub are supported via DOL. Therefore, DOL plays a central role in Ontohub's general architecture and feature set. For this reason we provide a short introduction to DOL in Section 3, which is followed by an overview of the architecture of Ontohub in Section 4. Further, in Section 5, we illustrate some of Ontohub's features. The examples presented there involve ontology alignments, the representation of competency questions, and Ontohub's theorem proving support. Finally, Section 6 concludes the paper by pointing to some important future work.

## 2. Design and features of Ontohub

Ontohub is a web-based semantic repository engine. Users of Ontohub can upload, browse, search and annotate basic ontologies in various languages via a web frontend.[4] Ontohub is open source under the GNU AGPL 3.0 license.[5] Currently, Ontohub has about 200 registered users, including ontology researchers and developers as well as Master and PhD students. In this section we discuss the design of Ontohub, provide an overview of its major features and compare it to existing work.

### 2.1. *Overall design of Ontohub*

The design of Ontohub was strongly influenced by the Open Ontology Repository (OOR) initiative (Baclawski and Schneider, 2009).[6] OOR is a long-term international effort, which established requirements and designed an overall architecture for ontology repositories:

> "[OOR aims at] promot[ing] the global use and sharing of ontologies by (i) establishing a hosted registry-repository; (ii) enabling and facilitating open, federated, collaborative ontology repositories, and (iii) establishing best practices for expressing interoperable ontology and taxonomy work

[3]This paper is an updated and extended version of Mossakowski et al. (2014). Among the new case studies presented here, we include one that appeared in Kuksa and Mossakowski (2016).

[4]See https://ontohub.org.

[5]The sources are available under https://github.com/ontohub/ontohub.

[6]See http://www.oor.net and http://ontologforum.org/index.php/OpenOntologyRepository.

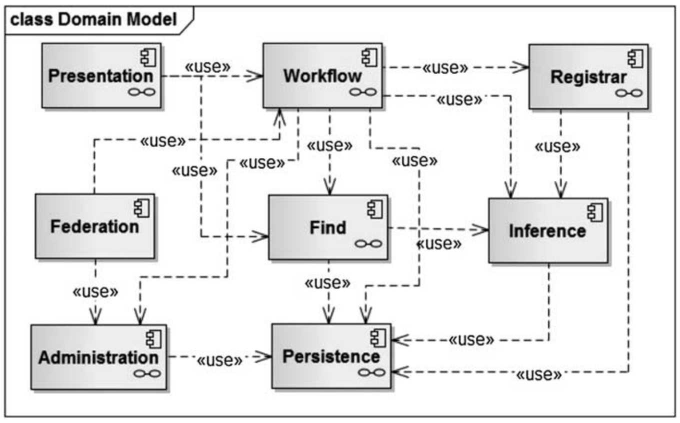


Fig. 1. Architecture of the Open Ontology Repository (OOR).

in registry-repositories, where an ontology repository is a facility where ontologies and related information artifacts can be stored, retrieved and managed." (Baclawski and Schneider, 2009)

One important goal of OOR is the support of ontology languages beyond OWL, for example Common Logic. Another goal is the support of logical inference. Moreover, OOR aims at a decentralised architecture, with different cooperating web services. This provides a highly modular design and supports the easier re-use and plug-in of components. OOR's requirements have been established[7] and a proposed architecture is shown in Fig. 1.[8] However, OOR is a long-term initiative, which does not yet have a complete implementation.[9]

Ontohub's overall design was conceived to satisfy a substantial subset of the requirements set out in the OOR initiative. Ontohub is the first implementation of a repository engine to meet these requirements, including an API for federation as well as support for logical inference and axiom selection. Moreover, it extends the initial OOR vision by several features suggested by the development of the DOL language. First and foremost, this includes abstracting particular ontology languages based on general model-theoretic semantics. This provides a principled logic-based support for heterogeneity in ontology design. This abstraction layer allows for a uniform treatment of different logics, both syntactically and semantically, and facilitates elegant and general solutions for ontology translation, structuring, aligment, and approaches to establish consistency.

A good sample use case for these novel capabilities is given by the recent FOUST initiative ('FOundational STance'),[10] an effort to build a digital archive hosted on Ontohub to provide authoritative formalized versions of the leading foundational ontologies, including the Basic Formal Ontology (BFO; Guizzardi and Wagner, 2004),[11] the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE; Borgo and Masolo, 2009),[12] the General Formal Ontology (GFO; Herre, 2010),[13] the Generalized Upper

[7]See http://ontologforum.org/index.php/OpenOntologyRepository_Requirement.
[8]See http://ontolog.cim3.net/wiki/OpenOntologyRepository_Architecture/Candidate03.html.
[9]The main implementation used by OOR is (a cosmetically adapted) BioPortal, which, however, does not follow the OOR principles to a significant extent. There are no OOR implementation efforts beyond Ontohub.
[10]See https://foust.inf.unibz.it.
[11]See https://github.com/BFO-ontology.
[12]See http://www.loa.istc.cnr.it/old/DOLCE.html.
[13]See http://www.onto-med.de/ontologies/gfo/.

Model (GUM; Bateman et al., 2010),[14] the Unified Foundational Ontology (UFO; Guizzardi and Wagner, 2004)[15] and Yet Another More Advanced Top-level Ontology (YAMATO; Borgo and Mizoguchi, 2014).[16] This will include variants of these ontologies defined in OWL and variants of first-order logic (FOL) and higher-order logic (HOL), as well as formally establishing their relationships based on theory interpretation or alignment. Another aim of the project is to provide consistency proofs, within Ontohub, by giving interpretations of the ontologies into formalised models.

## 2.2. *Features of Ontohub*

Based on the developed architecture, Ontohub offers the following central features:

**Multiple repositories:** ontologies can be organized in multiple repositories, each with its own management of editing and ownership rights.
**Git interface:** version control of ontologies is supported via interfacing the Git version control system.
**Linked-data compliance:** one and the same IRI is used for referencing an ontology, downloading it (for use with tools), and for user-friendly presentation in the browser.

Ontohub is unique in following OOR's ambitious goals, in particular:

**Modular architecture:** Ontohub is decoupled into different services.
**Multi-language support:** ontologies can be formalized in various logics like RDF, OWL, Common Logic, the Unified Modeling Language (UML),[17] the Common Algebraic Specification Language (CASL),[18] the input language of the TPTP Problem Library for Automated Theorem Proving (TPTP)[19] and higher-order logic.
**Logical inference:** intended consequences of ontologies can be proved.

Finally, Ontohub fully supports **modular and distributed ontologies** through DOL, a standard approved by the Object Management Group (OMG; Object Management Group, 2016; Mossakowski et al., 2013, 2015).[20] DOL provides a unified framework that supports ontologies formalized in more than one logic (i.e., heterogenous ontologies). Further, it supports a modular design of ontologies, where individual modules are connected by mappings. Examples of these mappings include alignments, interpretation of theories, conservative extensions, translation to other ontology languages etc. All of these features are equipped with a formal semantics. Since DOL plays such a central role in enabling features of Ontohub, we will discuss DOL in more detail in Section 3.

## 2.3. *Linked open data compliance*

In the RDF world, linked open data has become standard (Heath and Bizer, 2011; Auer et al., 2017). Linked open data is based on several principles: (1) data should be open source, (2) IRIs (generalised URLs) should be used to identify entities, such that (3) useful descriptions and representations can be

[14] See http://www.ontospace.uni-bremen.de/ontology/gum.html.
[15] See https://oxygen.informatik.tu-cottbus.de/drupal7/ufo/.
[16] See http://download.hozo.jp/onto_library/upperOnto.htm.
[17] See http://uml.org/.
[18] See http://www.cofi.info.
[19] See http://www.cs.miami.edu/~tptp/.
[20] See also dol-omg.org.

downloaded (via HTTP) at these IRIs (which therefore should be HTTP IRIs), (4) different formats are provided under the same IRI, depending on the MIME type of the HTTP request, and (5) relationships to other entities are provided. Altogether, these principles ensure that linked open data can be semantically queried.

However, in the OWL ontology world, to our knowledge, linked open data standards are generally not followed. While OWL ontologies often follow principles (1) and (2), (3) is not always followed, and (4) and (5) practically are never implemented. Usually, for download of an ontology in raw format and in HTML, one has to use different IRIs.

By contrast, in Ontohub, one and the same IRI is used for referencing an ontology, downloading it (for use with tools), and for user-friendly presentation in the browser. Depending on the MIME type of the request, under the IRI the raw ontology file is available, but also a HTML version for display in a browser and a Javascript Object Notation (JSON) version for processing with tools.

Ontohub eases the generation of such linked-data compliant IRIs: they are automatically generated through Ontohub's repository structure. Ontohub uses IRIs of the form https://ontohub.org/name-of-repository/path-within-repository.[21]

## 2.4. *Repository vs. repository engine*

While Ontohub's design is inspired by OOR, one major difference is that Ontohub is not a repository, but a repository engine. This means that Ontohub ontologies are organized into repositories. The screenshot in Fig. 2 shows a partial view of Ontohub's currently available repositories. Some of them, e.g. Bioportal or COLORE, are mirrors of repositories hosted elsewhere (as indicated with the mirror icons), while the others are native Ontohub repositories. The organisation into repositories has several advantages:

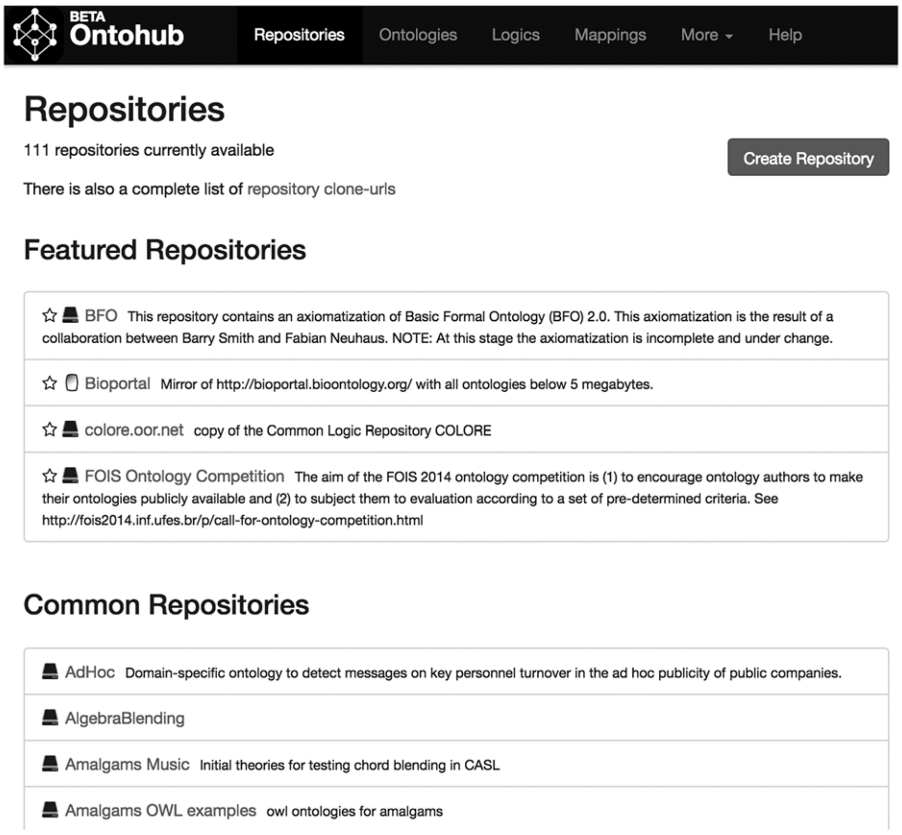
Firstly, repositories provide a certain structuring of ontologies, whether it be thematically or in terms of organisation. Access rights can be given to users or teams of users per repository. Typically, read access is given to everyone, and write access only to a restricted set of users and teams. However, also completely open world-writeable repositories are possible, as well as private repositories visible only to a restricted set of users and teams. Since creation of repositories is done easily with a few clicks, this supports a policy of many but small repositories (which of course does not preclude the existence of very large repositories). Note also that structuring within repositories is possible, since each repository is a complete file system tree.

Secondly, Ontohub repositories are Git repositories. Git is a popular decentralised version control system. With any Git client, the user can clone a repository to her local hard disk, edit it with any editor, and push the changes back to Ontohub. Alternatively, the web frontend can be used directly to edit ontologies; pushing will then be done automatically in the background. Parallel edits of the same file are synchronized and merged via Git; handling of merge conflicts can be done with Git merge tools.

Thirdly, ontologies can be searched globally in Ontohub, or in specific repositories. Additionally, user-supplied metadata can be used for searching.

For these reasons repositories are utilised as the main structuring elements of Ontohub. In this sense Ontohub is more similar to Github than to individual ontology repositories like Bioportal. We will compare Ontohub to Bioportal and other tools in more detail in the next section.

[21]In the future, this may change into https://ontohub.org/account-name/name-of-repository/path-within-repository.



Repositories

111 repositories currently available

There is also a complete list of repository clone-urls

Featured Repositories

BFO This repository contains an axiomatization of Basic Formal Ontology (BFO) 2.0. This axiomatization is the result of a collaboration between Barry Smith and Fabian Neuhaus. NOTE: At this stage the axiomatization is incomplete and under change.

Bioportal Mirror of http://bioportal.bioontology.org/ with all ontologies below 5 megabytes.

colore.oor.net copy of the Common Logic Repository COLORE

FOIS Ontology Competition The aim of the FOIS 2014 ontology competition is (1) to encourage ontology authors to make their ontologies publicly available and (2) to subject them to evaluation according to a set of pre-determined criteria. See http://fois2014.inf.ufes.br/p/call-for-ontology-competition.html

Common Repositories

AdHoc Domain-specific ontology to detect messages on key personnel turnover in the ad hoc publicity of public companies.

AlgebraBlending

Amalgams Music Initial theories for testing chord blending in CASL

Amalgams OWL examples owl ontologies for amalgams

Fig. 2. Overview of Ontohub repositories.

### 2.5. *Related work*

Apart from related, mostly theoretical work on modularisation and structuring (Cuenca Grau et al., 2008; Kutz et al., 2010; Khan and Keet, 2015), the most extensive conceptual work on ontology repository engines has been done by the Open Ontology Repository (OOR) initiative, which we discussed in detail in Section 2.1.

To sum-up and compare features of different approaches to designing ontology repositories, Table 1 presents an overview of several central Ontohub features in comparison with three other web-based ontology repository engines. BioPortal (Noy et al., 2009) is a repository engine that originates in the biomedical domain, but now has instances for various domains. Beyond browsing and searching, it provides means for annotating and aligning ontologies. Besides OWL, also related languages like the OBO flat file format (Mungall et al., 2012) are supported. WebProtégé (Tudorache et al., 2013) has been inspired by the well-known tool Protégé, but offers only part of Protégé's functionality. Although github is a repository engine not dedicated to ontologies, we have included it, because it is used by COLORE, a repository of Common Logic ontologies.

Table 1 compares various aspects of these four different repository engines. Version control, difference viewing and synchronous editing concern facilities to process the raw ontology file (as well as any other text files accompanying the Ontologies). Search is usually done over various ontologies or even repositories. The set of supported ontology languages (for expressing unstructured ontologies) greatly

Table 1
Comparison of features of ontology repository engines

| | Bioportal | Github | WebProtégé | Ontohub |
|---|---|---|---|---|
| version control | rudimentary | git | rudimentary | git |
| diff viewer | XML output | human readable | human readable | human readable |
| synchronous editing | yes | no | no | no |
| linked open data | no | no | no | yes |
| search | yes | yes (only full text) | yes | yes |
| supported languages | OWL, OBO, SKOS | none | OWL, RDF(S), Frames | OWL, OBO, RDF, Common Logic, TPTP/ FOL, THF, CASL, UML, modal logic, DOL and more |
| structuring | no | no | no | yes |
| alignment | yes | no | yes | yes |
| OWL metrics | yes | no | rudimentary | rudimentary |
| OWL visualisation | yes | no | yes | rudimentary |
| theorem proving | no | no | no | yes |
| metadata | categories, projects, groups, entry types | none | none | project, entry type, formality level |
| reviews | yes | no | no | no |
| additional features | – | bug tracker, wiki | discussions, watches, notifications | – |
| plugin architecture | no | yes | yes | no |

varies, as well as support for structuring and alignment of ontologies. Only Ontohub supports a larger variety of languages. Some metrics and visualisation for OWL are provided by some repository engines; the generalisation of this to more ontology languages is an interesting future problem. Theorem proving is supported by Ontohub only. Finally, there are features that reach beyond ontologies proper: metadata, reviews, bug trackers, wikis etc. Also, some repository engines (WebProtégé and github) have a plugin architecture. This would be also a useful feature for Ontohub.

## 3. The distributed ontology, model, and specification language

The development and maintenance of ontologies are expensive and time consuming. For these reasons it is imperative to *reuse existing ontologies*. However, there are often two challenges for reuse: the existing ontologies need to be *adapted to fit* their new purpose and there is a *lack of interoperability* between existing ontologies. For example, the adaption may require the extension of an ontology by new axioms, the extraction of a subset of the ontology, a change of semantics from open world to closed world, or the translation from one ontology language to another. The lack of interoperability is the consequence of the fact that ontology development is usually driven by project specific requirements. Thus, two ontologies that were developed independently will likely represent the same domain in different and often conflicting ways.

One of the major goals of Ontohub is to make the reuse of ontologies easier. This functionality is supported by its implementation of the Distributed Ontology, Model and Specification Language (DOL). DOL is not yet another ontology language, but a metalanguage that makes statements about ontologies, which has been standardised by the Object Management Group (OMG). DOL provides a framework, which simplifies ontology reuse by providing operations that both enable (a) the adaption of ontologies to new purposes and (b) the resolution of conflicts between ontologies that prevent interoperability. Some of the major functions of DOL include the following:

– DOL allows the reuse and annotation of ontologies in other ontology languages;
– DOL provides operations for defining new, structured ontologies based on existing ontologies (e.g., translations between ontology languages, the union of two ontologies, the renaming of terms, or the removal of axioms);
– DOL enables the definition of structural mappings between ontologies (e.g., ontology alignments);
– DOL enables the definition of normative connections between ontology (e.g., entailment between ontologies, or defining that an ontology is a conservative extension of another). These normative connections may be translated into proof obligations between ontologies.

DOL is itself a declarative language with a model-theoretic semantics. Since DOL is a meta-language that reuses existing ontologies, the semantics of a DOL-expression is typically derived from the semantics of one or more underlying ontologies (which may be written in more than one ontology languages) in combination with the semantics of the DOL operations. Thus, a major technical challenge for the development of the formal semantics of DOL (and indirectly for the implementation of DOL in Ontohub) was the need to specify DOL's semantics in a way that is neutral to a specific underlying ontology language.

It is beyond the scope of this paper to provide a detailed presentation of DOL. For this reason we will use the rest of this section for an informal explanation of how DOL and, thus, Ontohub is able to support heterogenous ontologies, that is ontologies that consist of components that are written in different ontology languages. We will illustrate the use of DOL within Ontohub with use cases in Section 5. A full specification of DOL is freely available (Object Management Group, 2016).

### *3.1. Ontology languages and logics*

As discussed in Section 2 one of the major goals of OOR and of the development of Ontohub is to provide a repository that supports a wide variety of ontology languages. On the most basic level that means that Ontohub is able to store and parse ontologies that are written in different serialisations (e.g., Turtle and RDF XML) or different languages (e.g., OWL 2 DL or Common Logic). More interestingly, the implementation of DOL enables Ontohub to support translations between ontology languages and even modular ontologies that consists of components that involve different ontology languages.

Ontohub is able to support these features, because most ontology languages are based on logics. Ontohub and DOL abstract from the differences between these logics using the concept of logic syntax, which we introduce below. This allows us to develop results independently of the particularities of a logical system. The main idea is to collect the non-logical symbols of the language in signatures and to assign to each signature the set of sentences that can be formed with its symbols. For each signature, we provide means for extracting the symbols it consists of, together with their kind. Signature morphisms are mappings between signatures. Formally, signatures and their morphisms form a so-called category, which can be understood as a graph together with a composition principle that identifies paths in the graph. Signature morphisms induce a mapping between the sentences of the signatures, usually by replacing the symbols that occur in a sentence with their image through the signature morphism. Example 3.1 illustrates how OWL is represented in this framework. Mossakowski et al. (2015) provides full details of the formal aspects of DOL.

**Example 3.1.** OWL signatures consist of sets of atomic classes, individuals and properties. OWL signature morphisms map classes to classes, individuals to individuals, and properties to properties. For an OWL signature $\Sigma$, sentences are subsumption relations between classes, membership assertions of individuals on classes and pairs of individuals in properties. Sentence translation along a signature morphism is simply replacement of non-logical symbols with their image along the morphism. The kinds of symbols are class, individual, object property and data property, respectively, and the set of symbols of a signature is the union of its sets of classes, individuals and properties.

A logic syntax can be complemented with a model theory, which introduces semantics for the language and gives a satisfaction relation between the models and the sentences of a signature. The result is an *institution* (Goguen and Burstall, 1992). Similarly, we can complement a logic syntax with a proof theory, introducing a derivability relation between sentences, thus obtaining an *entailment system* (Meseguer, 1989). In particular, this can be done for all logics in use in Ontohub.

There is also a notion of logic mapping, formalised as an institution comorphism (Goguen and Rosu, 2002). Logic mappings can be classified according to their accuracy (Mossakowski and Kutz, 2011). *Sublogics* are the most accurate mappings in that they are syntactic subsets. *Embeddings* come close to sublogics, like injective functions come close to subsets. A mapping can be *faithful* in the sense that logical consequence (or logical deduction) is preserved and reflected, that is, inference systems and engines for the target logic can be reused for the source logic (along the mapping). *(Weak) exactness* is a technical property that guarantees this faithfulness even in the presence of ontology structuring operations (Borzyszkowski, 2002).

## *3.2. A graph of logic translations*

Ontohub supports ontologies written in the ontology languages and logics shown in Fig. 3. A *logic* tab (see the Ontohub screenshot in Fig. 4) allows the display of ontologies sorted by their logic. As discussed above, DOL provides language constructs for the translation of an ontology along a logic translation (like those shown in Fig. 3). Since Ontohub supports DOL, these language constructs can also be used in Ontohub. Moreover, logic translations can also be used in Ontohub for theorem proving: proof goals can be translated to prover-supported logics. This is especially important for logics like Common Logic for which no native theorem prover exists. In order to apply this translation technique, the translation has to be faithful (see the discussion at the end of Section 3.1).

Figure 5 extends Fig. 3. It is a revised and extended version of the graph of logics and translations (Mossakowski and Kutz, 2011) and visualises the future plans for Ontohub. New nodes include UML class diagrams, OWL-Full (i.e. OWL with an RDF semantics instead of description logic semantics), and Common Logic without second-order features ($\mathsf{CL}^-$). We have defined the translations between all of these logics in earlier publications (Mossakowski et al., 2012; Mossakowski and Kutz, 2011). The definitions of the DOL-conformance of some central standard ontology languages and translations among them has been given as annexes to the standard, whereas the majority will be maintained in an open registry (cf. Section 3.2). Thus, the DOL standard is not limited to a fixed set of ontology languages. It will be possible to contribute new languages and mappings to the registry.

This concludes our discussion of DOL. We will illustrate its use in Ontohub by discussing three applications in Section 5.

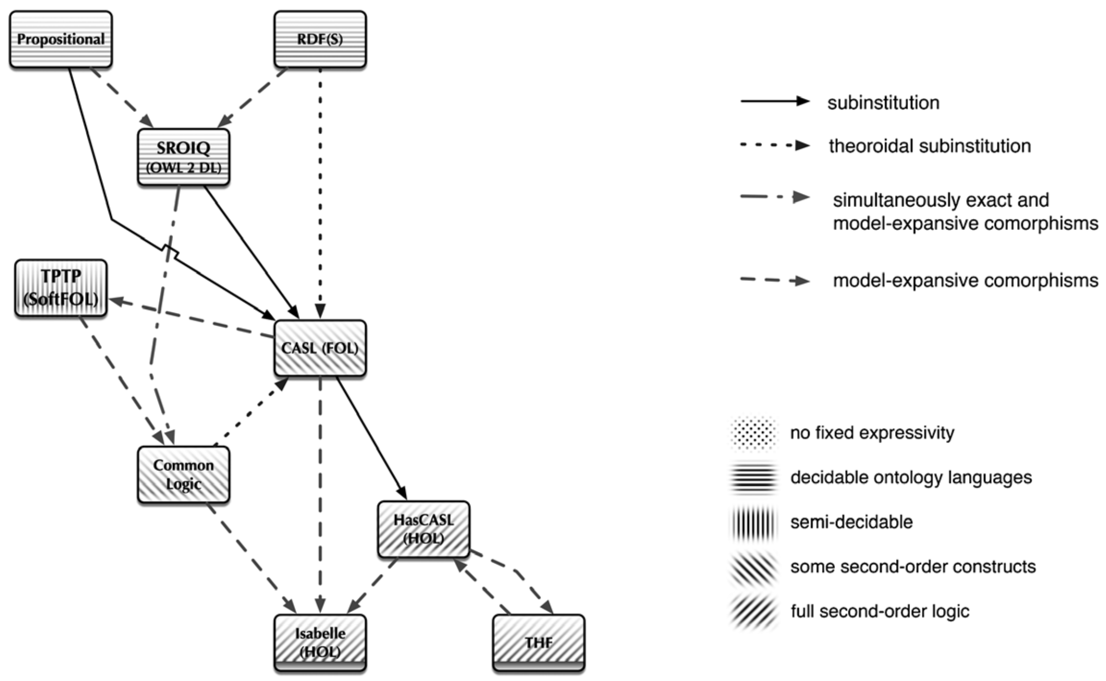
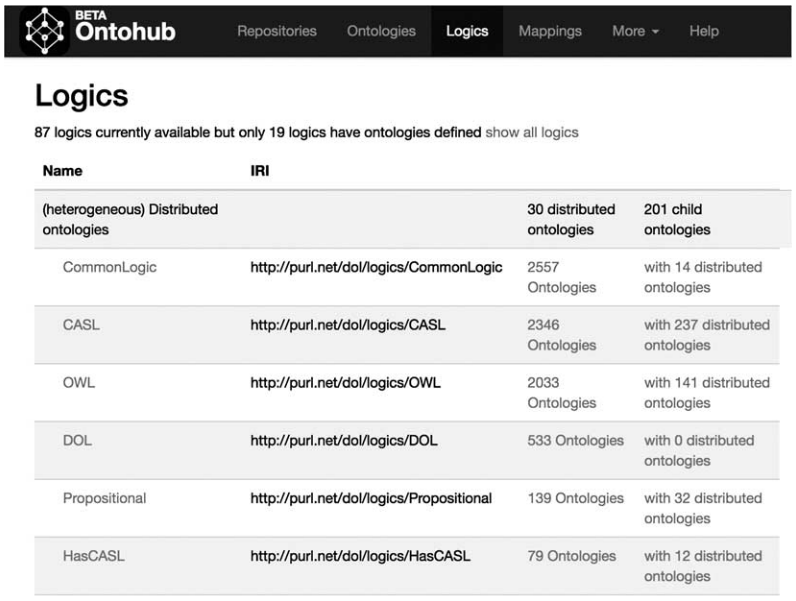


Fig. 3. The logic translation graph of currently Ontohub-supported languages.



## Logics

87 logics currently available but only 19 logics have ontologies defined show all logics

| Name | IRI | | |
|---|---|---|---|
| (heterogeneous) Distributed ontologies | | 30 distributed ontologies | 201 child ontologies |
| CommonLogic | http://purl.net/dol/logics/CommonLogic | 2557 Ontologies | with 14 distributed ontologies |
| CASL | http://purl.net/dol/logics/CASL | 2346 Ontologies | with 237 distributed ontologies |
| OWL | http://purl.net/dol/logics/OWL | 2033 Ontologies | with 141 distributed ontologies |
| DOL | http://purl.net/dol/logics/DOL | 533 Ontologies | with 0 distributed ontologies |
| Propositional | http://purl.net/dol/logics/Propositional | 139 Ontologies | with 32 distributed ontologies |
| HasCASL | http://purl.net/dol/logics/HasCASL | 79 Ontologies | with 12 distributed ontologies |

Fig. 4. ontohub.org portal: overview of logics.

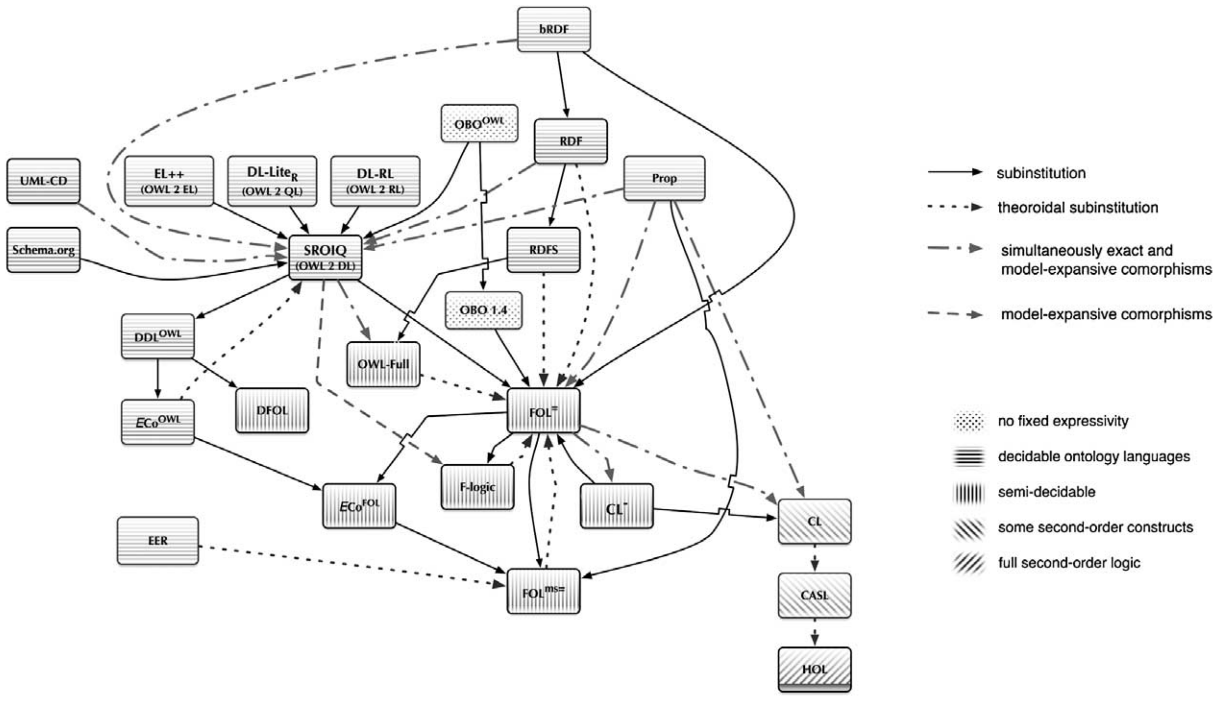


Fig. 5. The logic translation graph for DOL-conforming languages.

## 4. Architecture of Ontohub

Figure 6 depicts the Ontohub architecture. The most challenging part of Ontohub's implementation is the complex tool integration. The key feature of the OOR architecture (see Fig. 1) is the decoupling into decentralised services, which are ontologically described (thus arriving at Semantic Web services). With Ontohub, we are moving towards this architecture, while keeping a running and usable system. The OOR services depicted in Fig. 1 are visualised in the Ontohub architecture in Fig. 6: the large boxes with boldface inscriptions show how the OOR services in Fig. 1 are realised with Ontohub components. We now briefly describe these components and services.

The services are centrally integrated by the Ontohub *integration* layer, which is a Ruby on Rails application that also includes the *presentation* layer, i.e. a front-end providing the web interface, as well as the *administration* layer, i.e. user rights management and authorisation.

The *persistence* layer is based on Git (via git-svn, also Subversion repositories can be used) and an SQL database. The database backend is PostgreSQL. For the Git integration into the web application, a custom Git client was implemented in Ruby to be less prone to errors due to changes in new versions of the Git command line client. Efficient indexing and searching (the *find* layer) is done via Elasticsearch[22]), used with its official Ruby bindings.

A *federation* API allows the data exchange among different Ontohub and also with BioPortal instances. We therefore have generalised the OWL-based BioPortal API to arbitrary ontology languages, e.g. by abstracting classes and object properties to symbols of various kinds. Figure 6 indicates federation by showing several Ontohub and BioPortal instances.

[22]See https://www.elastic.co/products/elasticsearch.

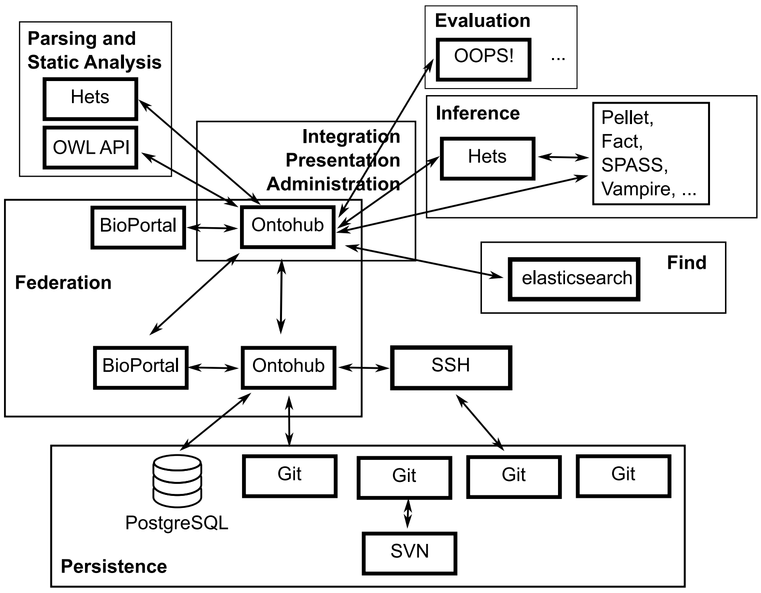


Fig. 6. Ontohub in a network of web services.

*Parsing and static analysis* is a RESTful service of its own provided by the Heterogeneous Tool Set (Hets; Mossakowski et al., 2007).[23] This task has two different aspects. Firstly, Hets is able to analyse a large number of basic ontology languages and logics; it returns the symbols and sentences of an ontology in JSON or XML format. While Hets can do this for a large variety of ontology languages, the OWL API does scale better for very large OWL ontologies. (The latter is an example of an Ontohub service that is provided for a restricted set of ontology languages.) Secondly, as discussed in Sections 2.1 and 3, DOL plays a central role for Ontohub, since it enables novel features, which ontology engineers may utilise to create new ontologies based on existing ontologies and define mappings between ontologies. The parsing and static analysis of DOL files is also handled by Hets.

We have integrated the OntOlogy Pitfall Scanner! (OOPS!; Poveda-Villalón et al., 2012) as an ontology *evaluation* service (for OWL only), and from the OOPS! API, we have derived a generalised API for use with other evaluation services.

*Inference* is done by encapsulating standard batch-processing reasoners (Pellet,[24] Fact++,[25] SPASS,[26] Vampire[27] etc.) into a RESTful API through Hets (which has been interfaced with 15 different reasoners). Integrating support for logical inference required a substantial extension of Hets's HTTP interface which returns proof details in JSON format. The prover-independent implementation of the Sumo Inference Engine (SInE) algorithm (Hoder and Voronkov, 2011) is a novelty in this field since it has only been used with few provers so far. In Ontohub it operates independently of the prover and, thus, supports any prover available in Ontohub.

[23] Available at http://hets.eu.
[24] See https://github.com/stardog-union/pellet.
[25] See http://owl.man.ac.uk/factplusplus.
[26] See http://spass-prover.org.
[27] See http://www.vprover.org.

## 5. Applications

In this section we will illustrate the novel capabilities of Ontohub by discussing three use cases: ontology alignment, the representation of competency questions, and Ontohub's integrated theorem proving capability.

### *5.1. Ontology alignment in Ontohub*

The foundational ontology (FO) repository Repository of Ontologies for MULtiple USes (ROMULUS)[28] contains alignments between a number of foundational ontologies, expressing semantic relations between the aligned entities. The alignments have been created manually and go beyond the capabilities of existing matching tools, as they rely on a deeper analysis of the involved concepts. We select three such ontologies, containing spatial and spatio-temporal concepts: DOLCE lite,[29] GFO[30] and BFO.[31] Our aim is to create an upper ontology of space, combining the knowledge in the three source ontologies. We present alignments between them using DOL syntax:[32]

```
%prefix(
           gfo: <http://www.onto-med.de/ontologies/>
           dolce: <http://www.loa-cnr.it/ontologies/>
           bfo: <http://www.ifomis.org/bfo/>

        )%
logic OWL

alignment DolceLite2BFO : dolce:DOLCE-Lite.owl to bfo:1.1 =
  endurant = IndependentContinuant,
  physical-endurant = MaterialEntity,
  physical-object = Object,                        perdurant = Occurrent,
  process = Process,                               quality = Quality,
  spatio-temporal-region = SpatiotemporalRegion,
  temporal-region = TemporalRegion,                space-region = SpatialRegion

alignment DolceLite2GFO : dolce:DOLCE-Lite.owl to gfo:gfo.owl =
  particular = Individual,             endurant = Presential,
  physical-object = Material_object, amount-of-matter = Amount_of_substrate,
  perdurant = Occurrent,           quality = Property,
  time-interval = Chronoid,            generic-dependent < necessary_for,
  part < abstract_has_part,            part-of < abstract_part_of,
  proper-part  <  has_proper_part,     proper-part-of  < proper_part_of,
  generic-location < occupies,         generic-location-of < occupied_by

alignment BFO2GFO : bfo:1.1 to gfo:gfo.owl =
  Entity = Entity,                           Object = Material_object,
  ObjectBoundary  = Material_boundary,       Role < Role,
  Occurrent = Occurrent, Process = Process, Quality = Property
  SpatialRegion  = Spatial_region,           TemporalRegion = Temporal_region
```

[28]See http://www.thezfiles.co.za/ROMULUS/home.html.
[29]See http://www.loa.istc.cnr.it/ontologies/DOLCE-Lite.owl.
[30]See http://www.onto-med.de/ontologies/gfo/.
[31]See http://www.ifomis.org/bfo/.
[32]This and the other examples from this section can be found at: https://ontohub.org/repositories/ontohubaopaperexamples.

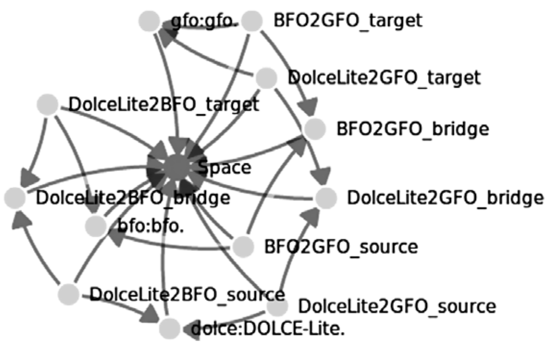


Fig. 7. Combination of ontologies along alignments.

$$\begin{tikzcd}
S & & A\_bridge & & T \\
& A\_source \arrow[ul] \arrow[ur] & & A\_target \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Fig. 8. Diagrams of DOL alignments.

We can then combine the ontologies while taking into account the semantic dependencies given by the alignments using DOL combinations:

```
ontology Space =
 combine BFO2GFO, DolceLite2GFO, DolceLite2BFO
```

The Ontohub screenshot in Fig. 7 shows the graph of links between ontologies created by Ontohub as a result of analysis of the `Space` ontology, which appears in the center of the graph. Around it and linking to it there are the aligned ontologies together with the diagrams resulting from the analysis of the alignments. The semantics of DOL alignments is given with the help of a W-shaped graph of ontologies and morphisms between them: if $A$ is an alignment between the ontologies $S$ and $T$, the diagram of $A$ is as in Fig. 8 where $A\_source$ and $A\_target$ contain the symbols from $S$ and $T$ (respectively) that appear in the correspondences of $A$ and $A\_bridge$ internalizes the semantics of the correspondences of $A$.[33] Thus in Fig. 7 we have the union of three such diagrams and links from each node of the union to its colimit ontology `Space`.

## 5.2. *Ontology competency questions with Ontohub and* DOL

'Competency questions' is the name for a methodology that supports behavior-driven development of ontologies. The approach can be, somewhat simplified, summarized in the following steps (Grüninger and Fox, 1995):

[33]We omit here an explanation of the ontology morphisms labeling the edges of the W-diagram. Details on the construction of this diagram and on semantics of DOL alignments can be found in Codescu et al. (2017).

1. The use cases for the soon-to-be-developed ontology are captured in the form of scenarios. Each scenario describes a possible state of the world and raises a set of competency questions. The answers to these competency questions should follow logically from the scenario – provided the knowledge that is supposed to be represented in the ontology.
2. A scenario and its competency questions are formalized or an existing formalization is refined.
3. The ontology is developed.
4. An automatic theorem prover is used to check whether the competency questions logically follow from the scenario and the ontology.
5. Steps (2–4) are repeated until all competency questions can be proven from the combination of the ontology and their respective scenarios.

Ontohub enables the representation and execution of competency questions with the help of DOL files. For example, assume the use case is to enable semantically enhanced searches for a database, which contains names of people, their gender, and information about parenthood. Let's assume we are planning to develop an ontology of family relationships called *familyRel.omn*. One way to capture the intended capabilities of the ontology is by providing a scenario and the competency question that the system should be able to answer in this scenario (see Fig. 9).

Ontohub supports the use of competency questions during ontology development in two ways. First, in Ontohub competency questions can be stored and maintained in the same repository as the ontologies that they refer to. Second, Ontohub provides automatic reasoning capabilities for many languages, which allows Ontohub to evaluate the ontology with the help of the competency questions.

The support of competency questions is realized in Ontohub with DOL. Technically, competency questions are formalized as part of heterogenous DOL ontologies, which consist of three parts: (i) the ontology-under-test; (ii) a formalization of the scenario, and (iii) a formalization of the competency questions. The DOL ontology treats each of the formulas in (iii) as conjectures that are supposed to be logically entailed of the union of (i) and (ii).

For example, Fig. 10 illustrates how the competency questions about family relationship ontology from the example above may be encoded in DOL. The DOL file refers to two external ontologies, namely the ontology-under-test `familyRelations` and the ontology `scenario`, which contains a representation of the facts of the scenario in Fig. 9. These ontologies are combined into the ontology `CQbase`

*In the scenario the following facts hold:*

- *Amy is female and a parent of Berta and Chris.*
- *Berta is female.*
- *Chris is male and a parent of Dora.*
- *Dora is female.*

*In this case the system should be able to answer the following questions*

- *Is Chris a father of Dora? (expected: yes)*
- *Is Berta a sister of Chris (expected: yes)*
- *Is Chris female? (expected: no)*
- *Is Amy older than Dora? (expected: yes)*

Fig. 9. Scenario and Competency Question.

```
%prefix(
 f1:   <http://ex.com/fr1#>
)%

logic OWL

ontology CQbase =
  <https://ontohub.org/appliedontologyontohubpaper/scenario>
     and
  <https://ontohub.org/appliedontologyontohubpaper/familyRelations>
end

%%  Is Chris a father? (expected: yes)
ontology chrisFather =  CQbase then %implies
  {  Individual: f1:Chris
       Types: Annotations: rdfs:label "ChrisIsAFather" f1:Father }
end

%%  Is Dora a child of Chris (expected: yes)
ontology doraChildChris =  CQbase then %implies
  {  Individual: f1:Dora
       Facts: Annotations: rdfs:label "DoraIsAChildOfChris" f1:child_of f1:Chris }
end

%%  Is Chris female? (expected: no)
ontology chrisFemale =  CQbase then %implies
  {  Individual: f1:Chris
       Types: Annotations: rdfs:label "ChrisIsNotFemale" not f1:Female }
end

%%  Is Amy older than Dora? (expected: yes)
ontology amyOlderDora =  CQbase then %implies
  {  Individual: f1:Amy
       Facts: Annotations: rdfs:label "AmyIsOlderThanDora" f1:older_than f1:Dora }
end
```

Fig. 10. Competency Questions for an OWL ontology represented in DOL.

with the help of DOL's "and" operator. In the following text each of the competency questions is represented in OWL Manchester Syntax, which is embedded in an DOL expression that expresses that the competency question is intended to be entailed by `CQbase`.

Ontohub analyses the DOL file and recognizes the competency questions as proof obligations that need to be met by `CQbase`, and, thus, indirectly by `familyRel`. The relationships (including the proof obligations) between the different ontologies are displayed by Ontohub graphically (see screen shot in Fig. 11.) These proof obligations correspond to conjectures that may be validated by Ontohub with the help of automatic theorem provers (see Section 5.3).

### 5.3. *Theorem proving in Ontohub*

Ontohub recognises proof obligations in ontologies, like the competency questions above, and allows the user to invoke automated theorem provers to attempt to prove these conjectures and thus turn them into theorems. For simplicity, open conjectures and proven theorems are both summarised under a "theorems" tab in the web application. Note that the term *conjecture* emphasises that the ontology is viewed

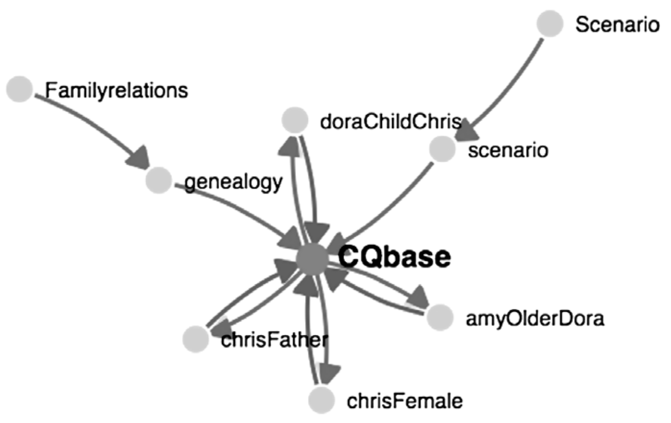
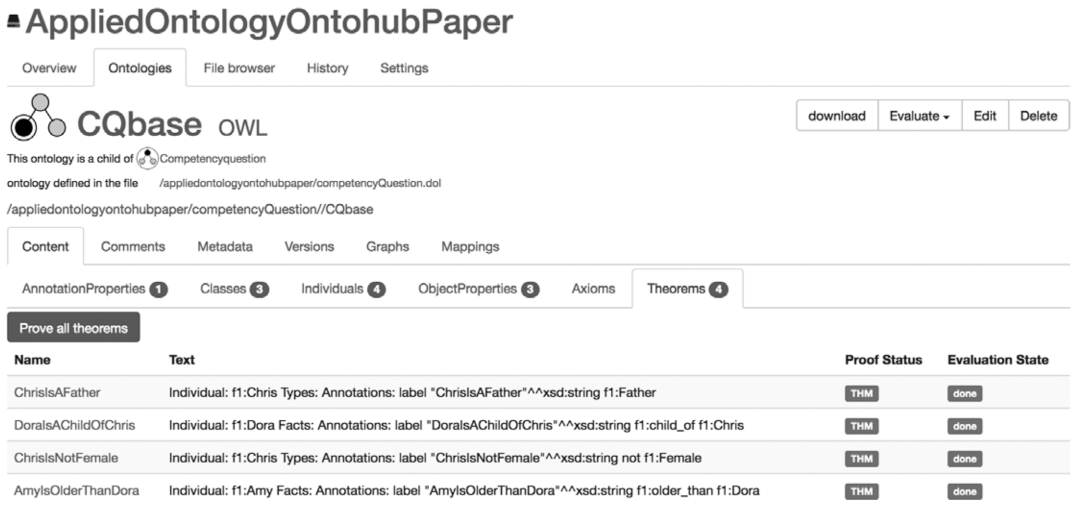


Fig. 11. Representation of the Competency Questions in Ontohub.



CQbase OWL

This ontology is a child of Competencyquestion

ontology defined in the file /appliedontologyontohubpaper/competencyQuestion.dol

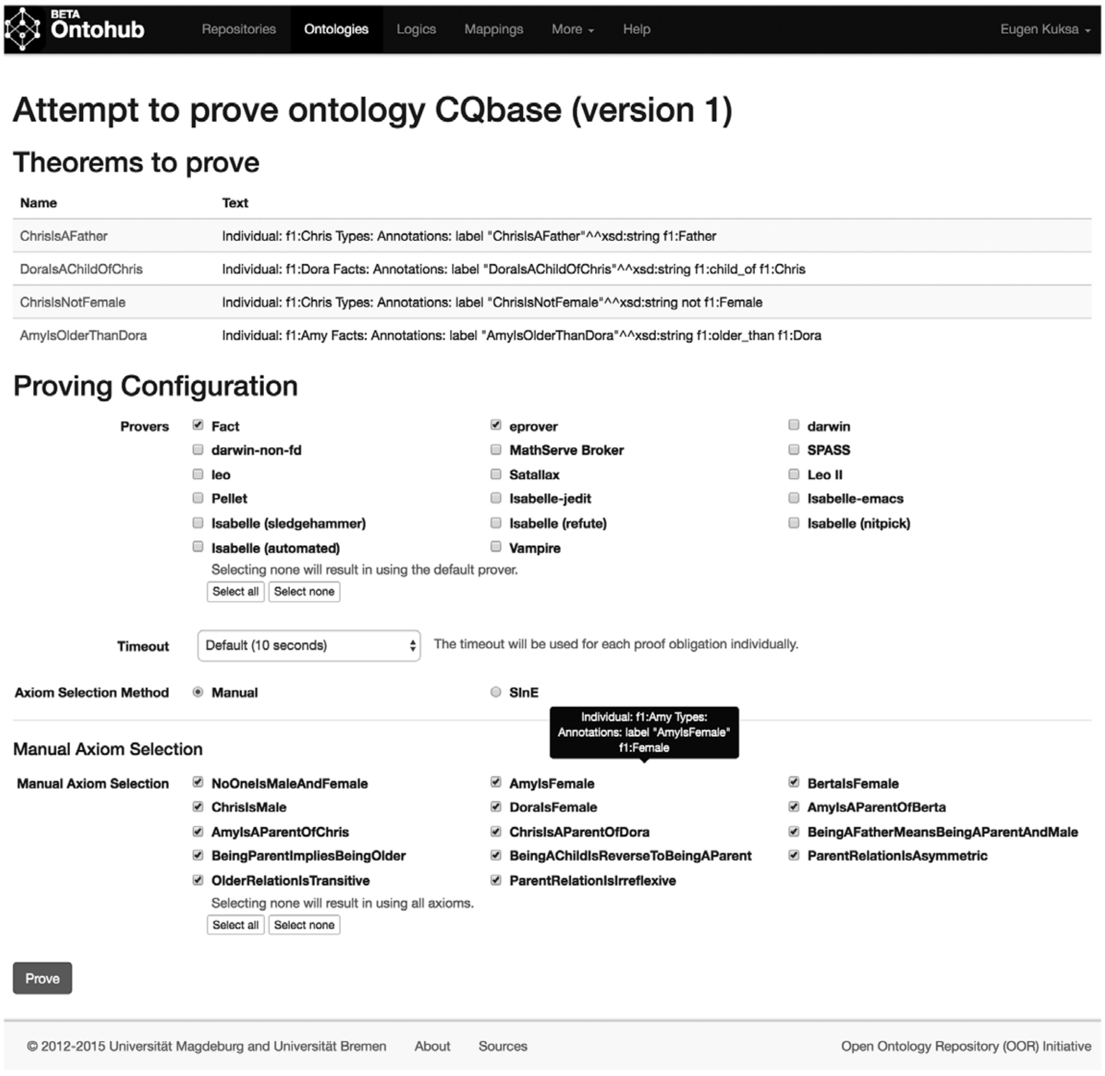
/appliedontologyontohubpaper/competencyQuestion//CQbase

| Name | Text | Proof Status | Evaluation State |
|---|---|---|---|
| ChrisIsAFather | Individual: f1:Chris Types: Annotations: label "ChrisIsAFather"^^xsd:string f1:Father | THM | done |
| DoraIsAChildOfChris | Individual: f1:Dora Facts: Annotations: label "DoraIsAChildOfChris"^^xsd:string f1:child_of f1:Chris | THM | done |
| ChrisIsNotFemale | Individual: f1:Chris Types: Annotations: label "ChrisIsNotFemale"^^xsd:string not f1:Female | THM | done |
| AmyIsOlderThanDora | Individual: f1:Amy Facts: Annotations: label "AmyIsOlderThanDora"^^xsd:string f1:older_than f1:Dora | THM | done |

Fig. 12. Overview of the statuses of all theorems.

as correct and fixed, and its logical consequences are to be discovered. From the perspective of competency questions, conjectures should be called *assertions*, because they assert that an ontology should imply certain logical consequences. So, the assertion is viewed as correct and fixed, while the ontology needs to be adapted if the consequence cannot be proved. In practice, often both the ontology and its intended consequences need to be corrected. Hence, we subsume both conjectures and assertions under the term *conjecture*.

When an ontology has been analysed, a "theorems" tab listing all conjectures of the ontology is shown under the "content" tab of the ontology page, as displayed in Fig. 12. There, the user can either choose to prove all conjectures at once or only a specific one. To perform such an action, the next step is to configure the proof attempts, as shown in Fig. 13. Above the actual configuration options, the selected "theorems" to be proved are listed with their names and their definitional text. This can be, for example, the competency questions of the previous section.



Attempt to prove ontology CQbase (version 1)

Theorems to prove

| Name | Text |
|---|---|
| ChrisIsAFather | Individual: f1:Chris Types: Annotations: label "ChrisIsAFather"^^xsd:string f1:Father |
| DoraIsAChildOfChris | Individual: f1:Dora Facts: Annotations: label "DoraIsAChildOfChris"^^xsd:string f1:child_of f1:Chris |
| ChrisIsNotFemale | Individual: f1:Chris Types: Annotations: label "ChrisIsNotFemale"^^xsd:string not f1:Female |
| AmyIsOlderThanDora | Individual: f1:Amy Facts: Annotations: label "AmyIsOlderThanDora"^^xsd:string f1:older_than f1:Dora |

Proving Configuration

Provers: [✓ Fact] [✓ eprover] darwin, darwin-non-fd, MathServe Broker, SPASS, leo, Satallax, Leo II, Pellet, Isabelle-jedit, Isabelle-emacs, Isabelle (sledgehammer), Isabelle (refute), Isabelle (nitpick), Isabelle (automated), Vampire
Selecting none will result in using the default prover.

Timeout: Default (10 seconds) The timeout will be used for each proof obligation individually.

Axiom Selection Method: [✓ Manual] SInE

Individual: f1:Amy Types: Annotations: label "AmyIsFemale" f1:Female

Manual Axiom Selection

Manual Axiom Selection: [✓ NoOneIsMaleAndFemale] [✓ AmyIsFemale] [✓ BertaIsFemale] [✓ ChrisIsMale] [✓ DoraIsFemale] [✓ AmyIsAParentOfBerta] [✓ AmyIsAParentOfChris] [✓ ChrisIsAParentOfDora] [✓ BeingAFatherMeansBeingAParentAndMale] [✓ BeingParentImpliesBeingOlder] [✓ BeingAChildIsReverseToBeingAParent] [✓ ParentRelationIsAsymmetric] [✓ OlderRelationIsTransitive] [✓ ParentRelationIsIrreflexive]
Selecting none will result in using all axioms.

© 2012-2015 Universität Magdeburg and Universität Bremen Open Ontology Repository (OOR) Initiative

Fig. 13. Configuration page for a proof attempt.

Multiple *provers* can be selected which are invoked in parallel for each selected conjecture. If no prover is actively selected by the user, a default prover chosen by Hets is used, e.g. SPASS for first-order logic.

A *timeout* for the automated theorem prover is used to limit the prover's resources. When the given amount of time is exceeded and no proof or refutation has been found yet, the prover is stopped. A different configuration may lead to a successful proof attempt.

*Axiom selection* can be used to restrict a proof attempt to only include a subset of the ontology's axioms for proving. This can reduce proving time and (in some cases) make proving feasible at all, which is particularly important for large ontologies. Axioms can be selected manually or automatically with a heuristic.

The manual method allows the user to select every axiom that may be needed for a proof individually. Figure 13 displays how the manual axiom selection is configured: For each axiom of the ontology, there is a check-box that allows the prover to use it. These check-boxes are labeled by the names of the axioms. Hovering the mouse over the check-box label reveals the definitional text of the axiom. Furthermore, the names and definitional texts of the axioms can be seen in the "axioms" tab of the ontology's "content" page. This manual approach can be useful for small ontologies, but can also result in tedious work for ontologies with many axioms.

The automatic heuristic is a prover-independent implementation of the SInE algorithm (Hoder and Voronkov, 2011). A very simplified view of the SInE algorithm is that it recursively selects axioms that share a symbol with the conjecture or with axioms that have been selected in previous recursion steps. It expects three parameters to be set by the user that influence how many axioms are selected. The configured axiom selection is shared between all the selected provers.

When a proof attempt is finished, it is assigned a proof status telling the result in a single word, as displayed in Fig. 12: For instance, the first conjecture (stating that Chris is a Father) has been evaluated with the resulting status "THM". These statuses are defined in the SZS ontology (Sutcliffe, 2008). The most common statuses used by provers are

(i) THM (Theorem): All models of the axioms are models of the conjecture. That is, a proof of the conjecture has been found.
(ii) CSA (CounterSatisfiable): Some models of the axioms are models of the conjecture's negation. That is, a counterexample has been found, and the conjecture has been disproved.
(iii) TMO (Timeout): The prover exceeded the time limit.

Of these statuses, "THM" and "CSA" indicate a successful prover run, while "TMO" shows that the prover did not finish successfully by exceeding the given amount of time. We extended the SZS ontology[34] by a status specifically for proving with reduced axiom sets:

(iv) CSAS (CounterSatisfiableWithSubset): Some models of the selected axioms are models of the conjecture's negation ("A counterexample has been found, but not all axioms have been used").

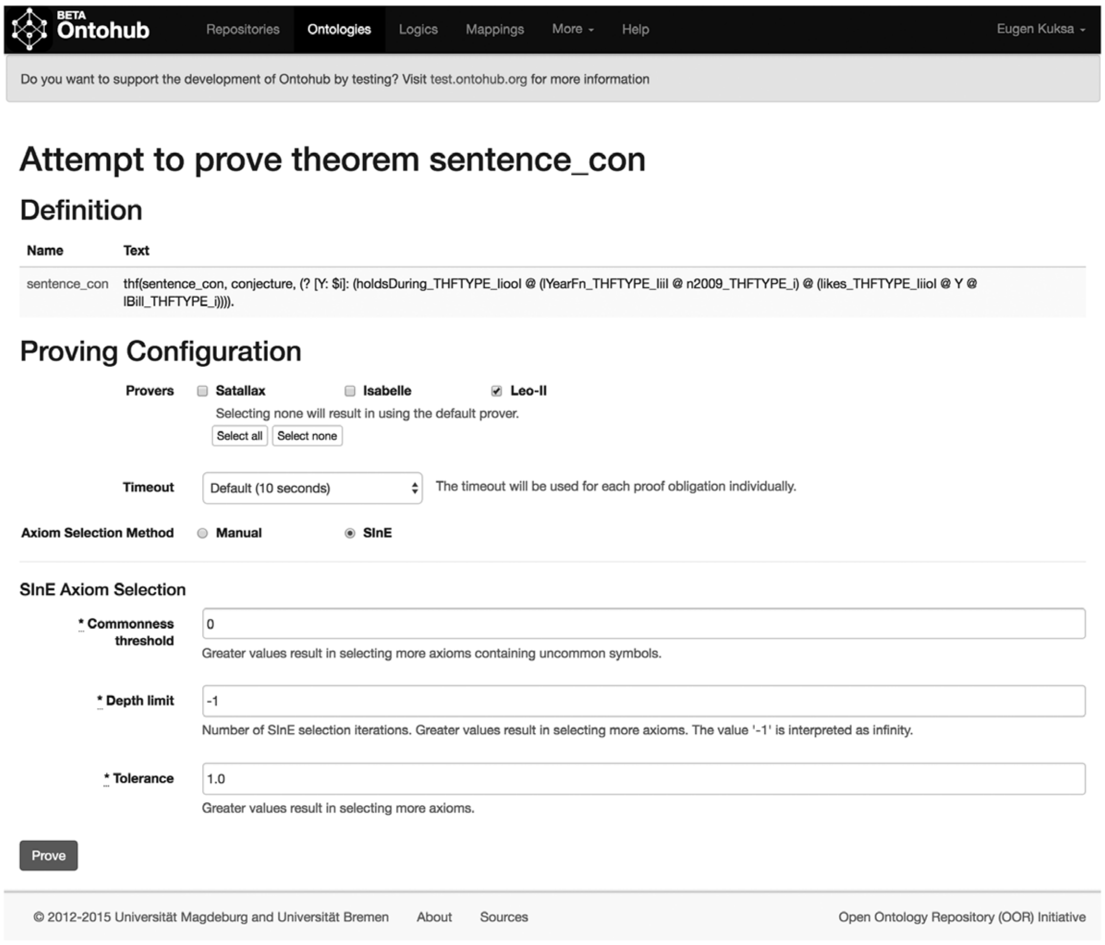
If a refutation of the conjecture is found using a strict subset of the axioms (which means that the prover returns with "CSAS"), we do not know whether the conjecture is really false or we have excluded an axiom that is crucial to a potentially existing proof. If the prover returns with "THM", by monotonicity of entailment relations, we know that the found proof is also a proof with the full axiom set because the same proof steps are valid nevertheless.[35]

Details of each proof attempt, including the proof itself if the attempt finished, can be inspected on the proof attempts page. There, the user can see, for example, the configuration with the selected axioms and the axioms that were used.

While the example "CQbase" describes a workflow of ontology development, it is too small to reveal the benefits of axiom selection. Therefore, we show a second example, concerning the Suggested Upper Merged Ontology (SUMO; Niles and Pease, 2001). It has been extracted from the TPTP library v6.4.0 (Sutcliffe, 2009), namely problem "CSR005^0" (it is named "sumo_ problem" in the figures). It contains over 5000 axioms and one conjecture. If a proof attempt is started with the configuration shown in Fig. 14, which is the default configuration for SInE, only 430 axioms are selected for proving and the proof is found within a second. The details of this proof attempt are displayed in Fig. 15. If, on the other hand, all axioms are selected, the prover runs into a timeout and returns with no result. These two proof

[34]Our modified SZS ontology can be found on http://ontohub.org/meta/proof_statuses.

[35]Note that this also holds for inconsistent ontologies: from them, everything can be proved – even though some OWL provers will return an error when an inconsistent ontology is presented to them.



Do you want to support the development of Ontohub by testing? Visit test.ontohub.org for more information

Attempt to prove theorem sentence_con

Definition

| Name | Text |
|---|---|
| sentence_con | thf(sentence_con, conjecture, (? [Y: $i]: (holdsDuring_THFTYPE_liool @ (lYearFn_THFTYPE_liil @ n2009_THFTYPE_i) @ (likes_THFTYPE_liiol @ Y @ lBill_THFTYPE_i)))). |

Proving Configuration

Provers: Satallax [✓ Leo-II] Isabelle
Selecting none will result in using the default prover.
Select all | Select none

Timeout: Default (10 seconds) — The timeout will be used for each proof obligation individually.

Axiom Selection Method: Manual [✓ SInE]

SInE Axiom Selection

* Commonness threshold: 0
Greater values result in selecting more axioms containing uncommon symbols.

* Depth limit: -1
Number of SInE selection iterations. Greater values result in selecting more axioms. The value '-1' is interpreted as infinity.

* Tolerance: 1.0
Greater values result in selecting more axioms.

Prove

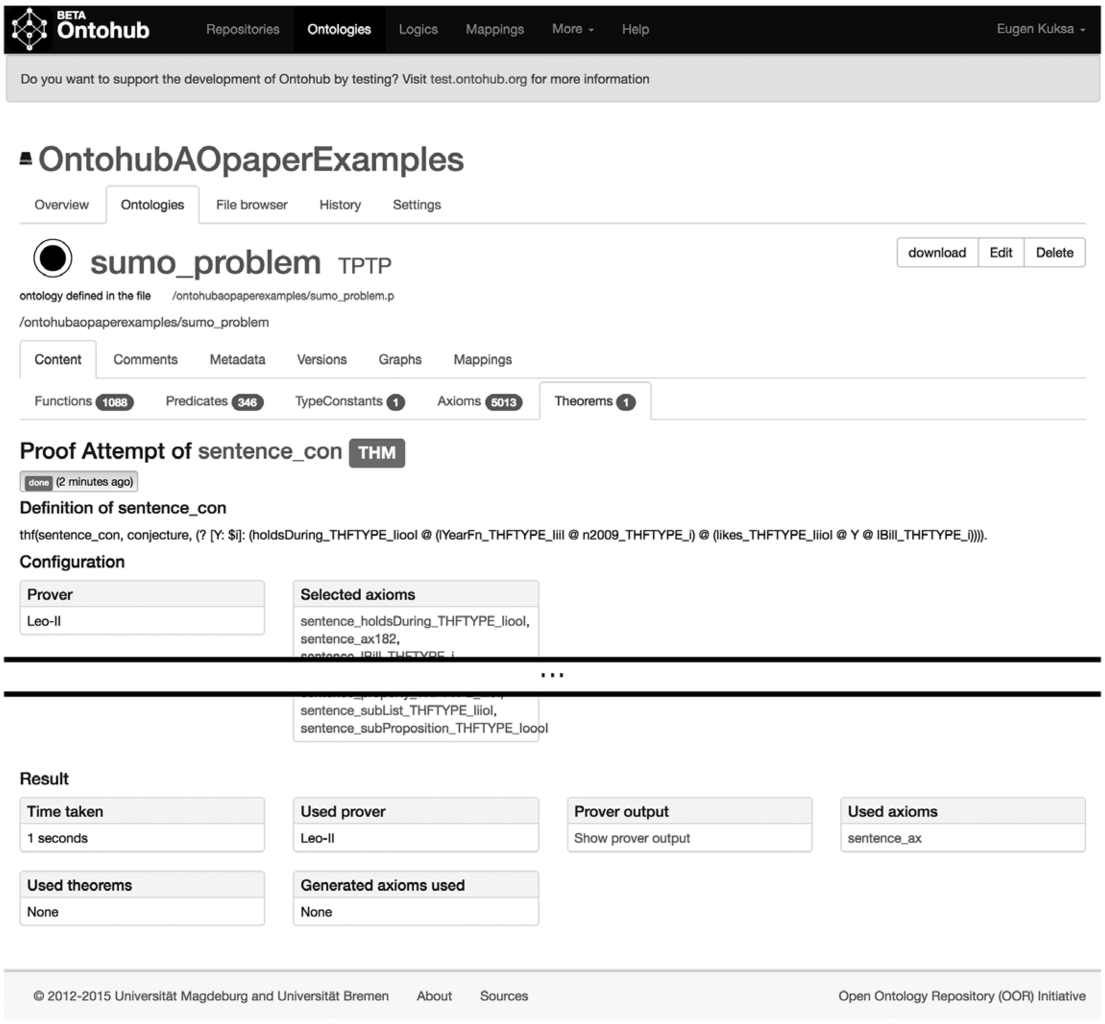
© 2012-2015 Universität Magdeburg and Universität Bremen   About   Sources   Open Ontology Repository (OOR) Initiative

Fig. 14. Configuration of SInE for a theorem proving problem from SUMO.

attempts are shown in Fig. 16, where the attempt with SInE has the number 1 and the attempt without SInE (using all axioms) has the number 2. This is one of many examples of the positive effect of axiom selection in theorem proving.

## 6. Conclusions and future work

Ontohub is a repository engine that is being developed with the needs of the ontology community in mind. Ontohub has features similar to a general purpose web-based Git repository engine. Users may create their own Git repositories to store, publish, and manage their ontologies. However, in addition Ontohub provides many ontology-specific services. It is linked-data compliant, it is able to analyse ontologies in all major ontology languages, and it provides automatic theorem proving capability for many of them. Further, Ontohub's support for DOL unlocks novel capabilities. We have illustrated some of them in Section 5, where we showed how to represent ontology alignments in Ontohub and how to formalise competency questions and use Ontohub's theorem proving capabilities to evaluate them.



Do you want to support the development of Ontohub by testing? Visit test.ontohub.org for more information

OntohubAOpaperExamples

sumo_problem TPTP

ontology defined in the file /ontohubaopaperexamples/sumo_problem.p

/ontohubaopaperexamples/sumo_problem

Proof Attempt of sentence_con THM

done (2 minutes ago)

Definition of sentence_con

thf(sentence_con, conjecture, (? [Y: $i]: (holdsDuring_THFTYPE_liool @ (lYearFn_THFTYPE_liil @ n2009_THFTYPE_i) @ (likes_THFTYPE_liiol @ Y @ lBill_THFTYPE_i)))).

Configuration

| Prover | Selected axioms |
|---|---|
| Leo-II | sentence_holdsDuring_THFTYPE_liool, sentence_ax182, sentence_lBill_THFTYPE_i … sentence_subList_THFTYPE_liiol, sentence_subProposition_THFTYPE_loool |

Result

| Time taken | Used prover | Prover output | Used axioms |
|---|---|---|---|
| 1 seconds | Leo-II | Show prover output | sentence_ax |

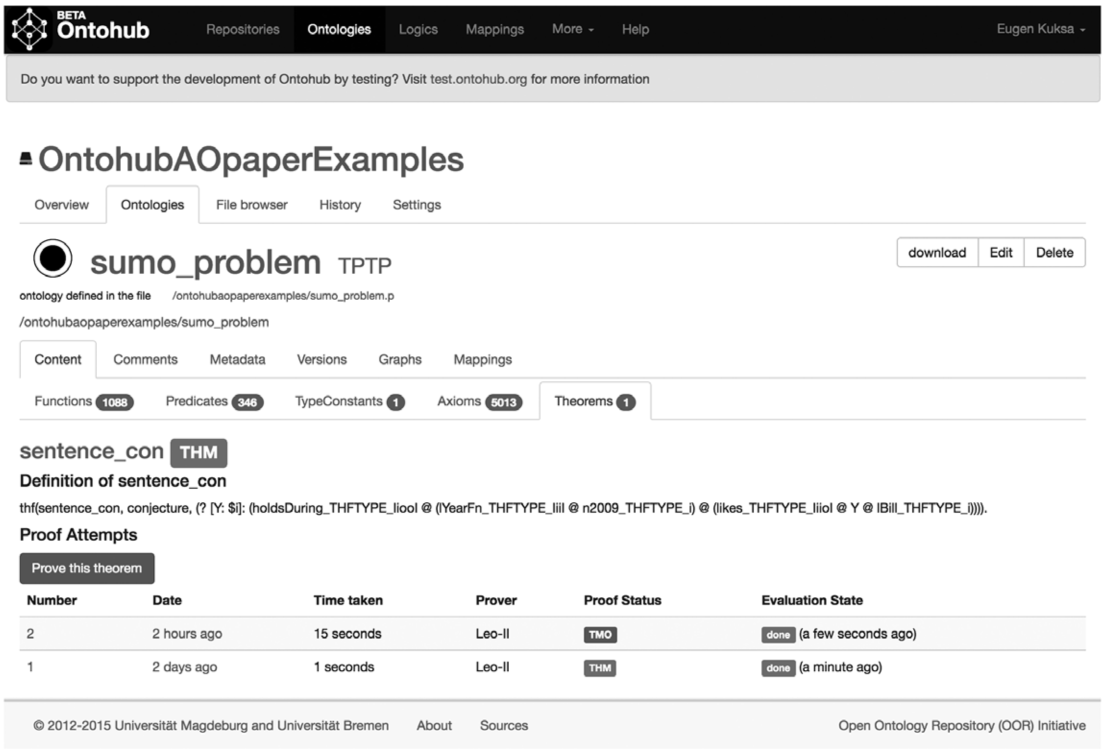
| Used theorems | Generated axioms used |
|---|---|
| None | None |

© 2012-2015 Universität Magdeburg and Universität Bremen About Sources Open Ontology Repository (OOR) Initiative

Fig. 15. Details of the finished proof attempt with SInE.

Ontohub is on its way from a research prototype to productive use. For example, the FOIS 2014 ontology competition has used Ontohub as platform for uploading ontologies used in submissions.[36] Ontologies used in FOIS papers often need expressiveness beyond OWL; here, the multi-logic nature of Ontohub is essential. Similarly, Ontohub has played an important role in the interdisplipinary EU FP7 Project COINVENT 'Concept Invention Theory'[37] where Ontohub repositories were used to store logical theories in a variety of logical languages (FOL, Common Logic, HOL, OWL) and from different domains such as from ontology, music, and mathematics (Schorlemmer et al., 2014). Finally, the recent FOUST initiative ('FOundational STance')[38] will use Ontohub for creating a digital archive with the leading foundational ontologies (including BFO, DOLCE, GFO, GUM, UFO and YAMATO, see Section 2.1 for references).

[36]See https://ontohub.org/fois-ontology-competition.
[37]See http://www.coinvent-project.eu.
[38]See https://foust.inf.unibz.it.



Do you want to support the development of Ontohub by testing? Visit test.ontohub.org for more information

OntohubAOpaperExamples

sumo_problem TPTP

ontology defined in the file /ontohubaopaperexamples/sumo_problem.p

/ontohubaopaperexamples/sumo_problem

sentence_con THM

Definition of sentence_con

```
thf(sentence_con, conjecture, (? [Y: $i]: (holdsDuring_THFTYPE_liool @ (lYearFn_THFTYPE_liil @ n2009_THFTYPE_i) @ (likes_THFTYPE_liiol @ Y @ lBill_THFTYPE_i)))).
```

Proof Attempts

| Number | Date | Time taken | Prover | Proof Status | Evaluation State |
|---|---|---|---|---|---|
| 2 | 2 hours ago | 15 seconds | Leo-II | TMO | done (a few seconds ago) |
| 1 | 2 days ago | 1 seconds | Leo-II | THM | done (a minute ago) |

© 2012-2015 Universität Magdeburg and Universität Bremen About Sources Open Ontology Repository (OOR) Initiative

Fig. 16. Overview of the proof attempts on the SUMO problem. Proof attempt 1 is with SInE, proof attempt 2 is without SInE.

Future work will improve stability and useability, and include the completion of full DOL support and the integration of ontology evaluation and workflow tools. The integration of interactive provers bears many challenges. A first step is the integration of the Isabelle theorem prover via the web interface Clide (Lüth and Ring, 2013), which is currently equipped with an API for this purpose.

Currently, a re-implementation of Ontohub is under way. The goal is to advance the splitting of the infrastructure into different services. Most prominently, we plan to implement the Ontohub frontend completely in React/Javascript, while Ruby on Rails is only used for the JSON API served by the back-end. Communication with Hets, which is currently done by the background process manager Sidekiq,[39] will be done in the future through RabbitMQ,[40] which is a fully-fledged service broker. This will ease the distribution of Hets parsing and proving services across server farms.

## References

Auer, S., Berners-Lee, T., Bizer, C., Capadisli, S., Heath, T., Janowicz, K. & Lehmann, J. (Eds.) (2017). *Workshop on Linked Data on the Web Co-Located with 26th International World Wide Web Conference (WWW 2017), CEUR Workshop Proceedings*. CEUR-WS.org.

Baclawski, K. & Schneider, T. (2009). The open ontology repository initiative: Requirements and research challenges. In *Proceedings of Workshop on Collaborative Construction, Management and Linking of Structured Knowledge at the ISWC* (p. 18).

[39]See http://sidekiq.org/.

[40]See https://www.rabbitmq.com/.

Bateman, J.A., Hois, J., Ross, R. & Tenbrink, T. (2010). A linguistic ontology of space for natural language processing. *Artificial Intelligence*, *174*(14), 1027–1071. doi:10.1016/j.artint.2010.05.008.

Borgo, S. & Masolo, C. (2009). Foundational choices in DOLCE. In S. Staab and R. Studer (Eds.), *Handbook on Ontologies*. International Handbooks on Information Systems (pp. 361–381). Springer.

Borgo, S. & Mizoguchi, R. (2014). A first-order formalization of event, object, process and role in YAMATO. In P. Garbacz and O. Kutz (Eds.), *Formal Ontology in Information Systems – Proceedings of the Eighth International Conference, FOIS 2014*, September, 22–25, 2014, Rio de Janeiro, Brazil. Frontiers in Artificial Intelligence and Applications (Vol. 267, pp. 79–92). IOS Press.

Borzyszkowski, T. (2002). Logical systems for structured specifications. *Theoretical Computer Science*, *286*, 197–245. doi:10.1016/S0304-3975(01)00317-6.

Codescu, M., Mossakowski, T. & Kutz, O. (2017). A categorical approach to networks of aligned ontologies. *Journal on Data Semantics*. To appear.

Cuenca Grau, B., Horrocks, I., Kazakov, Y. & Sattler, U. (2008). Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research (JAIR)*, *31*, 273–318.

Goguen, J. & Rosu, G. (2002). Institution morphisms. *Formal aspects of computing*, *13*, 274–307. doi:10.1007/s001650200013.

Goguen, J.A. & Burstall, R.M. (1992). Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, *39*, 95–146. Predecessor in: *LNCS*, Vol. 164, pp. 221–256, 1984.

Grüninger, M. & Fox, M.S. (1995). Methodology for the design and evaluation of ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI.95*, Montreal.

Guizzardi, G. & Wagner, G. (2004). A unified foundational ontology and some applications of it in business modeling. In J. Grundspenkis and M. Kirikova (Eds.), *Knowledge and Model Driven Information Systems Engineering for Networked Organisations*, CAiSE'04 Workshops in Connection with the 16th Conference on Advanced Information Systems Engineering, Proceedings, 7–11 June, 2004, Riga, Latvia (Vol. 3, pp. 129–143). Riga, Latvia: Faculty of Computer Science and Information Technology, Riga Technical University.

Heath, T. & Bizer, C. (2011). *Linked Data: Evolving the Web Into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology* (1st ed.). Morgan & Claypool Publishers.

Herre, H. (2010). General Formal Ontology (GFO): A foundational ontology for conceptual modelling. In R. Poli and L. Obrst (Eds.), *Theory and Applications of Ontology* (Vol. 2). Berlin: Springer.

Hoder, K. & Voronkov, A. (2011). Sine Qua Non for large theory reasoning. In *CADE 23* (pp. 299–314).

Khan, Z.C. & Keet, C.M. (2015). An empirically-based framework for ontology modularisation. *Applied Ontology*, *10*(3–4), 171–195. doi:10.3233/AO-150151.

Kuksa, E. & Mossakowski, T. (2016). Ontohub: Version control, linked data and theorem proving for ontologies. In O. Kutz et al. (Eds.), *Proceedings of the Joint Ontology Workshops (JOWO-2016). Episode 2: The French Summer of Ontology, Co-Located with the 9th International Conference on Formal Ontology in Information Systems (FOIS 2016)*, July 6–9, 2016, Annecy, France. CEUR Workshop Proceedings (Vol. 1660, pp. 6–9). CEUR-WS.org.

Kutz, O., Mossakowski, T. & Lücke, D. (2010). Carnap, Goguen, and the hyperontologies: Logical pluralism and heterogeneous structuring in ontology design. *Logica Universalis*, *4*(2). Special issue on 'Is Logic Universal?'. doi:10.1007/s11787-010-0020-3.

Lüth, C. & Ring, M. (2013). A web interface for Isabelle: The next generation. In *Intelligent Computer Mathematics* (pp. 326–329). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-39320-4_22.

Manola, F. & Miller, E. (2004). RDF Primer. W3C Recommendation. World Wide Web Consortium (W3C).

Meseguer, J. (1989). General logics. In H.J. Ebbinghaus (Ed.), *Logic Colloquium '87* (pp. 275–329). North Holland.

Mossakowski, T., Codescu, M., Neuhaus, F. & Kutz, O. (2015). The distributed ontology, modelling and specification language – DOL. In A. Koslow and A. Buchsbaum (Eds.), *The Road to Universal Logic–Festschrift for 50th Birthday of Jean-Yves Beziau, Vol. II*. Studies in Universal Logic (pp. 489–520). Birkhäuser.

Mossakowski, T. & Kutz, O. (2011). The onto-logical translation graph. In O. Kutz and T. Schneider (Eds.), *Modular Ontologies*. Frontiers in Artificial Intelligence and Applications (Vol. 230, pp. 94–109). IOS Press.

Mossakowski, T., Kutz, O. & Codescu, M. (2014). Ontohub: A semantic repository for heterogeneous ontologies. In *Proc. of the Theory Day in Computer Science (DACS-2014), Satellite Workshop of ICTAC-2014*, September 15–16, 2014. University of Bucharest.

Mossakowski, T., Kutz, O., Codescu, M. & Lange, C. (2013). The distributed ontology, modeling and specification language. In C.D. Vescovo et al. (Eds.), *WoMO-13* (Vol. 1081). CEUR-WS.

Mossakowski, T., Lange, C. & Kutz, O. (2012). Three semantics for the core of the distributed ontology language. In M. Donnelly and G. Guizzardi (Eds.), *7th International Conference on Formal Ontology in Information Systems (FOIS)*. Frontiers in Artificial Intelligence and Applications (Vol. 239, pp. 337–352). IOS Press. FOIS Best Paper Award.

Mossakowski, T., Maeder, C. & Lüttich, K. (2007). The heterogeneous tool set. In O. Grumberg and M. Huth (Eds.), *TACAS 2007*. Lecture Notes in Computer Science (Vol. 4424, pp. 519–522). Heidelberg: Springer.

Mungall, C., Ruttenberg, A., Horrocks, I. & Osumi-Sutherland, D. (2012). Obo flat file format 1.4 syntax and semantics.

Niles, I. & Pease, A. (2001). Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems* (Vol. 2001, pp. 2–9). ACM.

Noy, N., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.-A., Chute, C.G., et al. (2009). Bioportal: Ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, *37*(2), W170–W173.

Object Management Group (2016). The distributed ontology, modeling. and specification language (DOL). OMG standard. Available at: http://www.omg.org/spec/DOL.

OWL Working Group (2009). OWL 2 web ontology language: Document overview. W3C recommendation. World Wide Web Consortium (W3C).

Poveda-Villalón, M., Suárez-Figueroa, M.C. & Gómez-Pérez, A. (2012). Validating ontologies with OOPS! In *Knowledge Engineering and Knowledge Management* (pp. 267–281). Springer. doi:10.1007/978-3-642-33876-2_24.

Schorlemmer, M., Smaill, A., Kühnberger, K.-U., Kutz, O., Colton, S., Cambouropoulos, E. & Pease, A. (2014). COINVENT: Towards a computational concept invention theory. In *Proc. of the 5th International Conference on Computational Creativity*, Ljubljana, Slovenia.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., et al. (2007). The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, *25*(11), 1251–1255. doi:10.1038/nbt1346.

Sutcliffe, G. (2008). The SZS ontologies for automated reasoning software. In G. Sutcliffe, P. Rudnicki, R. Schmidt, B. Konev and S. Schulz (Eds.), *Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and the 7th International Workshop on the Implementation of Logics*. CEUR Workshop Proceedings (Vol. 418, pp. 38–49).

Sutcliffe, G. (2009). The TPTP problem library and associated infrastructure: The FOF and CNF parts, v3.5.0. *Journal of Automated Reasoning*, *43*(4), 337–362. doi:10.1007/s10817-009-9143-8.

Tudorache, T., Nyulas, C., Noy, N.F. & Musen, M.A. (2013). Webprotégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic Web*, *4*(1), 89–99.