

# Ontohub

## A semantic repository for heterogeneous ontologies

Till Mossakowski, Oliver Kutz, and Mihai Codescu

Institute of Knowledge and Language Engineering  
Otto-von-Guericke University of Magdeburg, Germany

**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. It supports a wide range of formal logical and ontology languages building on the OntoIOP.org project and allows for complex inter-theory (concept) mappings and relationships with formal semantics.

Ontohub aims at satisfying a subset of the requirements for an Open Ontology Repository (OOR). OOR is a long-term international initiative, which has not resulted in a complete implementation so far, but established requirements and designed an architecture. Furthermore, Ontohub is being developed in close connection with the Distributed Ontology Language, which is going to be submitted as response to the Object Management Group's Ontology, Model and Specification Integration and Interoperability (OntoIOP) Request For Proposal.

## 1 Introduction

Ontologies play a central role for enriching data with a conceptual semantics and hence form an important backbone of the Semantic Web. Now the number of ontologies that are being built or already in use is steadily growing. This means that facilities for organizing ontologies into repositories, searching, maintenance and so on are becoming more important.

Existing ontology search engines and repositories include search engines like Swoogle, Watson, and Sindice. They concentrate on (full-text and structured) search and querying. TONES [1] is a repository for OWL [8] ontologies that provides some metrics, as well as an OWL sublanguage analysis. BioPortal [20] is a repository that originates in the biomedical domain, but now has instances for various domains. Beyond browsing and searching, it provides means for commenting and aligning ontologies. Besides OWL, also related languages like OBO [22] are supported. The NeOn Toolkit [2] supports searching, selecting, comparing, transforming, aligning and integrating ontologies. Besides OWL, also F-logic [11] is supported. Ontohub's design and architecture are intended to consolidate the strengths of these various repositories (e.g. support for specific languages, support for mappings, social semantic web features, etc.) in one uniform framework while at the same time addressing their shortcomings (e.g. lack of formal semantics).

Namely, Ontohub enjoys the following distinctive features:

- ontologies can be organized in multiple repositories, each with its own management of editing and ownership rights,
- private repositories are possible,
- version control of ontologies is supported via interfacing the Git version control system,
- ontologies can be edited both via the browser and locally with any editor (and in the latter case pushed via Git); Git will synchronize both editing approaches,
- one and the same URL is used for referencing an ontology, downloading it (for use with tools), and for user-friendly presentation in the browser (i.e. Ontohub is fully linked-data compliant)
- modular and distributed ontologies are specially supported,
- ontologies can not only be aligned (as in BioPortal and NeOn), but also be combined along alignments,
- logical relations between ontologies (interpretation of theories, conservative extensions etc.) are supported,
- support for a variety of ontology languages, in particular OWL, RDF, Common Logic [9], first-order logic, and relational database schemes; (in preparation are: UML, F-logic, distributed description logics,  $\mathcal{E}$ -connections),
- ontologies can be translated to other ontology languages, and compared with ontologies in other languages,
- heterogeneous ontologies involving several languages can be built,
- ontology languages and ontology language translations are first-class citizens and are available as linked data.

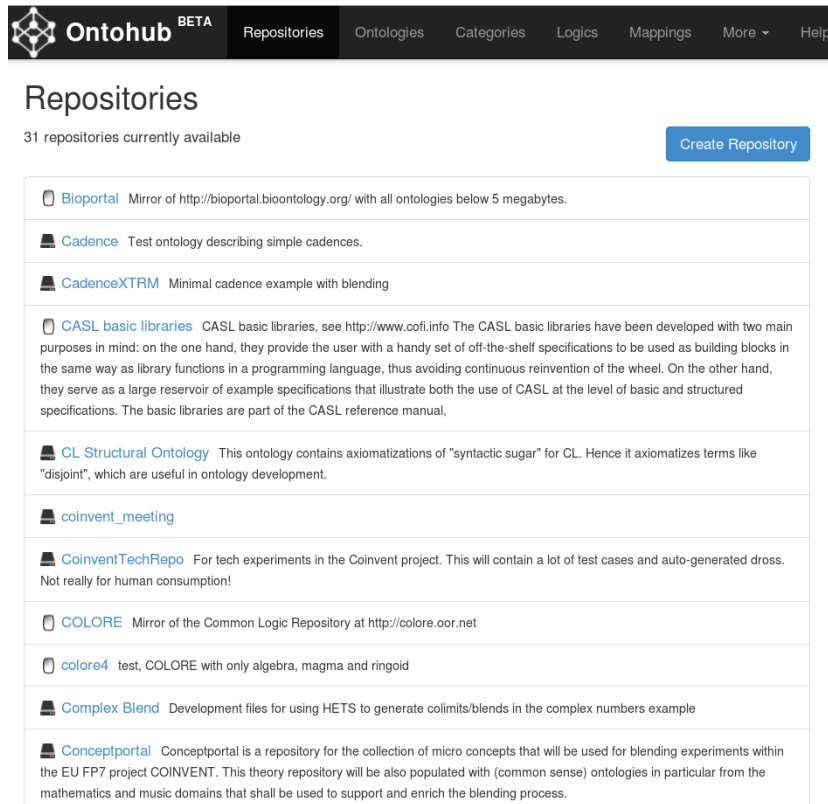
Ontohub’s central means for achieving this generality is based on the theoretical foundations underlying the distributed ontology language (DOL), introduced in Section 3 below.

Users of Ontohub can upload, browse, search and annotate basic ontologies in various languages via a web frontend, see <https://ontohub.org>. Ontohub is open source under GNU AGPL 3.0 license, the sources are available at the following URL <https://github.com/ontohub/ontohub>.

## 2 Ontohub: a linked-data compliant repository engine

Ontohub is not a repository, but a semantic repository engine. This means that Ontohub ontologies are organized into repositories. See Fig. 1 for an overview of the currently available repositories. Some of them, e.g. Bioportal or COL-ORE, are mirrors of repositories hosted elsewhere (as indicated with the mirror items), while the others are native Ontohub repositories. The organisation into repositories has several advantages:

- Firstly, repositories provide a certain structuring of ontologies, let it be thematically or organisational. Access rights can be given to users or teams of users per repository. Typically, read access is given to everyone, and write














**Ontohub** BETA

Repositories   Ontologies   Categories   Logics   Mappings   More ▾   Help

## Repositories

31 repositories currently available Create Repository

 <b>Bioportal</b> Mirror of <a href="http://bioportal.bioontology.org/">http://bioportal.bioontology.org/</a> with all ontologies below 5 megabytes.
 <b>Cadence</b> Test ontology describing simple cadences.
 <b>CadenceXTRM</b> Minimal cadence example with blending
 <b>CASL basic libraries</b> CASL basic libraries, see <a href="http://www.cofi.info">http://www.cofi.info</a> The CASL basic libraries have been developed with two main purposes in mind: on the one hand, they provide the user with a handy set of off-the-shelf specifications to be used as building blocks in the same way as library functions in a programming language, thus avoiding continuous reinvention of the wheel. On the other hand, they serve as a large reservoir of example specifications that illustrate both the use of CASL at the level of basic and structured specifications. The basic libraries are part of the CASL reference manual.
 <b>CL Structural Ontology</b> This ontology contains axiomatizations of "syntactic sugar" for CL. Hence it axiomatizes terms like "disjoint", which are useful in ontology development.
 <b>coinvent_meeting</b>
 <b>CoinventTechRepo</b> For tech experiments in the Coinvent project. This will contain a lot of test cases and auto-generated dross. Not really for human consumption!
 <b>COLORE</b> Mirror of the Common Logic Repository at <a href="http://colore.oor.net">http://colore.oor.net</a>
 <b>colore4</b> test, COLORE with only algebra, magma and ringoid
 <b>Complex Blend</b> Development files for using HETS to generate colimits/blends in the complex numbers example
 <b>Conceptportal</b> Conceptportal is a repository for the collection of micro concepts that will be used for blending experiments within the EU FP7 project COINVENT. This theory repository will be also populated with (common sense) ontologies in particular from the mathematics and music domains that shall be used to support and enrich the blending process.

**Fig. 1.** Overview of Ontohub repositories

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 that also structuring within repositories is possible, since each repository is a complete file system tree.

- Secondly, 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 like categories, formality levels and purposes can be used for searching.

Ontohub is linked-data compliant. This means that ontologies are referenced by a unique URL of the form `https://ontohub.org/name-of-repository/path-within-repository`. Depending on the MIME type of the request, under this URL, the raw ontology file will be available, but also a HTML version for display in a browser, an XML and a JSON version for processing with tools.

### 3 The Distributed Ontology Language (DOL) – Overview

Name	IRI		
(heterogeneous) Distributed Ontologies		2 distributed Ontologies	6 child Ontologies
OWL	<a href="http://purl.net/dol/logics/OWL">http://purl.net/dol/logics/OWL</a>	1192 Ontologies	with 29 distributed Ontologies
CASL	<a href="http://purl.net/dol/logics/CASL">http://purl.net/dol/logics/CASL</a>	687 Ontologies	with 16 distributed Ontologies
Propositional	<a href="http://purl.net/dol/logics/Propositional">http://purl.net/dol/logics/Propositional</a>	1 Ontology	with 0 distributed Ontologies
SoftFOL	<a href="http://purl.net/dol/logics/SoftFOL">http://purl.net/dol/logics/SoftFOL</a>	340 Ontologies	with 0 distributed Ontologies
CommonLogic	<a href="http://purl.net/dol/logics/CommonLogic">http://purl.net/dol/logics/CommonLogic</a>	557 Ontologies	with 0 distributed Ontologies
DOL	<a href="http://purl.net/dol/logics/DOL">http://purl.net/dol/logics/DOL</a>	1983 Ontologies	with 0 distributed Ontologies

Fig. 2. ontohub.org portal: overview of logics

The modularity mechanisms of Ontohub are based on those of the Distributed Ontology Language (DOL). DOL aims at providing a unified framework for (1) ontologies formalized in heterogeneous logics, (2) modular ontologies, (3) links between ontologies, and (4) annotation of ontologies.

An ontology in the Distributed Ontology Language (DOL) consists of modules formalized in *basic ontology languages*, such as OWL (based on description logic) or Common Logic (based on first-order logic with some

second-order features). These modules are serialized in the existing syntaxes of these languages in order to facilitate reuse of existing ontologies. DOL adds a meta-level on top, which allows for expressing heterogeneous ontologies and links between ontologies.<sup>1</sup> Such links include (heterogeneous) *imports* and *alignments*, *conservative extensions* (important for the study of ontology modules), and *theory interpretations* (important for reusing proofs). Thus, DOL gives ontology interoperability a formal grounding and makes heterogeneous ontologies and services based on them amenable to automated verification. The basic syntax and semantics of DOL can be found in [18, 17], and the general theory of heterogeneous specifications for ontologies in [12]. DOL uses internationalized resource identifiers (IRIs) for all its entities in order to foster linked data compliance.

<sup>1</sup> The languages that we call “basic” ontology languages here are usually limited to one logic and do not provide meta-theoretical constructs.

### 3.1 Case study: ontology alignment in Ontohub

The foundational ontology (FO) repository Repository of Ontologies for MULTiple USes (ROMULUS)<sup>2</sup> contains alignments between a number of foundational ontologies, expressing semantic relations between the aligned entities. We select three such ontologies, containing spatial and temporal concepts: DOLCE<sup>3</sup>, GFO<sup>4</sup> and BFO<sup>5</sup>, and present alignments between them using DOL syntax:

```
%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
```

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

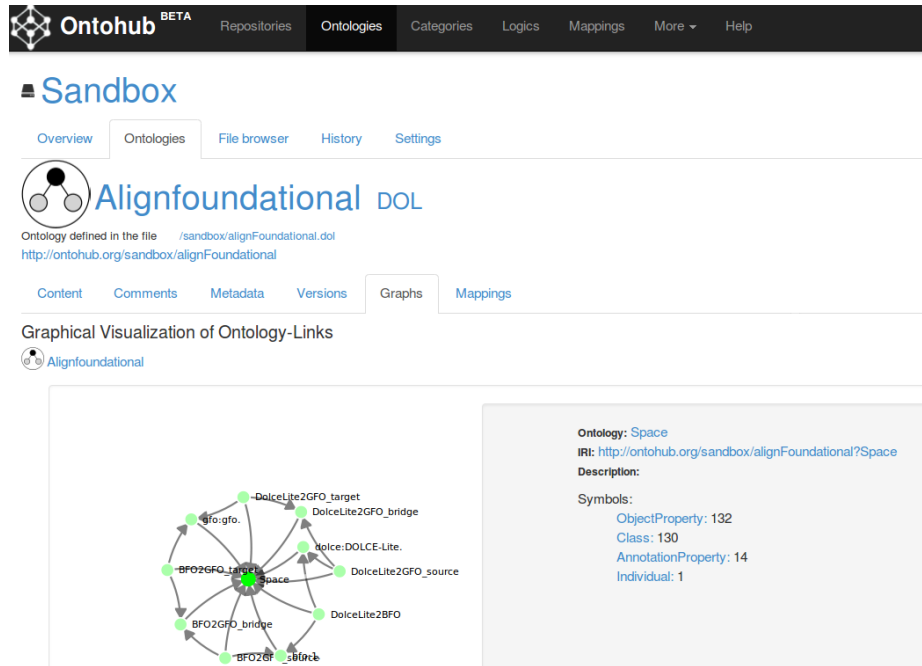
<sup>2</sup> See <http://www.thezfiles.co.za/ROMULUS/home.html>

<sup>3</sup> See <http://www.loa.istc.cnr.it/DOLCE.html>

<sup>4</sup> See <http://www.onto-med.de/ontologies/gfo/>

<sup>5</sup> See <http://www.ifomis.org/bfo/>

**ontology Space =**  
**combine** BF02GF0, DolceLite2GF0, DolceLite2BFO



**Fig. 3.** Combination of ontologies along alignments.

Fig. 3 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.<sup>6</sup>

## 4 The Distributed Ontology Language (DOL) – Foundations

### 4.1 Logics

The large variety of logics in use can be captured at an abstract level 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

<sup>6</sup> Details on the construction of these diagrams can be found in [?].

signature, we provide means for extracting the symbols it consists of, together with their kind. Signature morphisms are mappings between signatures. We do not assume any details except that signature morphisms can be composed and there are identity morphisms; this amounts to a category of signatures. Readers unfamiliar with category theory may replace this with a partial order (signature morphisms are then just inclusions). See [17] for details of this simplified foundation.

**Definition 1.** A logic syntax  $L = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Symbols}, \mathbf{Kinds}, \mathbf{Sym}, \mathbf{kind})$  consists of

- a category **Sign** of signatures and signature morphisms;
- a sentence functor<sup>7</sup>  $\mathbf{Sen} : \mathbf{Sign} \rightarrow \mathbf{Set}$  assigning to each signature the set of its sentences and to each signature morphism  $\sigma : \Sigma \rightarrow \Sigma'$  a sentence translation function  $\mathbf{Sen}(\sigma) : \mathbf{Sen}(\Sigma) \rightarrow \mathbf{Sen}(\Sigma')$ ;
- a set **Symbols** of symbols and a set **Kinds** of symbol kinds together with a function  $\mathbf{kind} : \mathbf{Symbols} \rightarrow \mathbf{Kinds}$  giving the kind of each symbol;
- a functor  $\mathbf{Sym} : \mathbf{Sign} \rightarrow \mathbf{Set}$  assigning to each signature  $\Sigma$  a set of symbols  $\mathbf{Sym}(\Sigma) \subseteq \mathbf{Symbols}$ .

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 a so-called *institution* [5]. Similarly, we can complement a logic syntax with a proof theory, introducing a derivability relation between sentences, thus obtaining an *entailment system* [15]. In particular, this can be done for all logics in use in Ontohub.

*Example 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.

In this framework, an ontology  $O$  over a logic syntax  $L$  is a pair  $(\Sigma, E)$  where  $\Sigma$  is a signature and  $E$  is a set of  $\Sigma$ -sentences. Given an ontology  $O$ , we denote by  $\mathbf{Sig}(O)$  the signature of the ontology. An ontology morphism  $\sigma : (\Sigma_1, E_1) \rightarrow (\Sigma_2, E_2)$  is a signature morphism  $\sigma : \Sigma_1 \rightarrow \Sigma_2$  such that  $\sigma(E_1)$  is a logical consequence of  $E_2$ . Several notions of *translations* between logics can be introduced. In the case of logic syntaxes, the simplest variant of translation from  $L_1$  to  $L_2$  maps  $L_1$ -signatures to  $L_2$ -signatures along a functor  $\Phi$  and  $\Sigma$ -sentences in  $L_1$  to  $\Phi(\Sigma)$ -sentences in  $L_2$ , for each  $L_1$ -signature  $\Sigma$ , in a compatible way with the sentence translations along morphisms. The complexity of translation

<sup>7</sup> If running between partial orders, a functor is just a mapping.

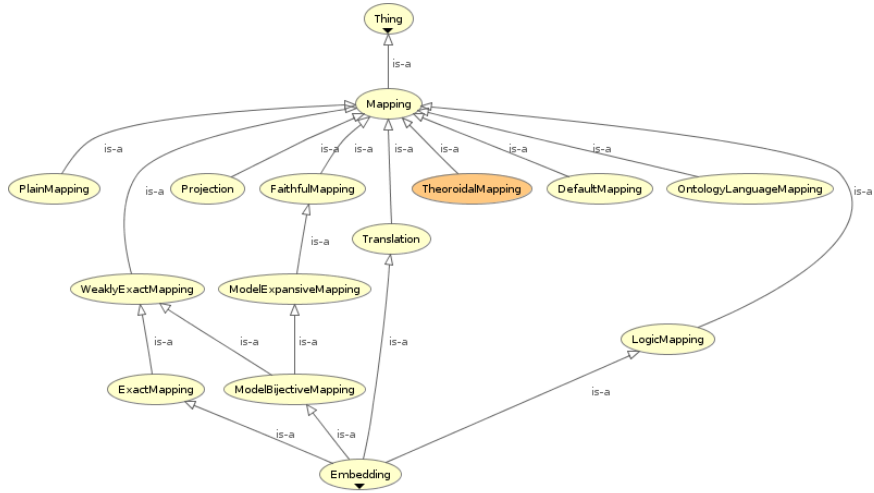


Fig. 4. The part of the LoLa ontology concerning mappings

increases when a model theory or a proof theory is added to the logic syntax. Fig. 4 shows the inferred class hierarchy below the class `Mapping` of the LoLa ontology (see Sect. 4.3 below), as computed within PROTÉGÉ. Mappings are split along the following dichotomies:

- *translation* versus *projection*: a translation embeds or encodes a logic into another one, while a projection is a forgetful operation (e.g. the projection from first-order logic to propositional logic forgets predicates with arity greater than zero). Technically, the distinction is that between institution comorphisms and morphisms [6].
- *plain mapping* versus *simple theoroidal mapping* [6]: while a plain mapping needs to map signatures to signatures, a simple theoroidal mapping maps signatures to theories. The latter therefore allows for using “infrastructure axioms”: e.g. when mapping OWL to Common Logic, it is convenient to rely on a first-order axiomatization of a transitivity predicate for properties etc.

Mappings can also be classified according to their accuracy, see [16] for details. *Sublogics* are the most accurate mappings: they are just 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 presences of ontology structuring operations [3].

## 4.2 A Graph of Logic Translations

Fig. 5 is a revised and extended version of the graph of logics and translations introduced in [16]. 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 ( $CL^-$ ). We have defined the translations between all of these logics in earlier publications [18, 16]. The definitions of the DOL-conformance of some central standard ontology languages and translations among them will be given as annexes to the standard, whereas the majority will be maintained in an open registry (cf. Sec. 4.3).

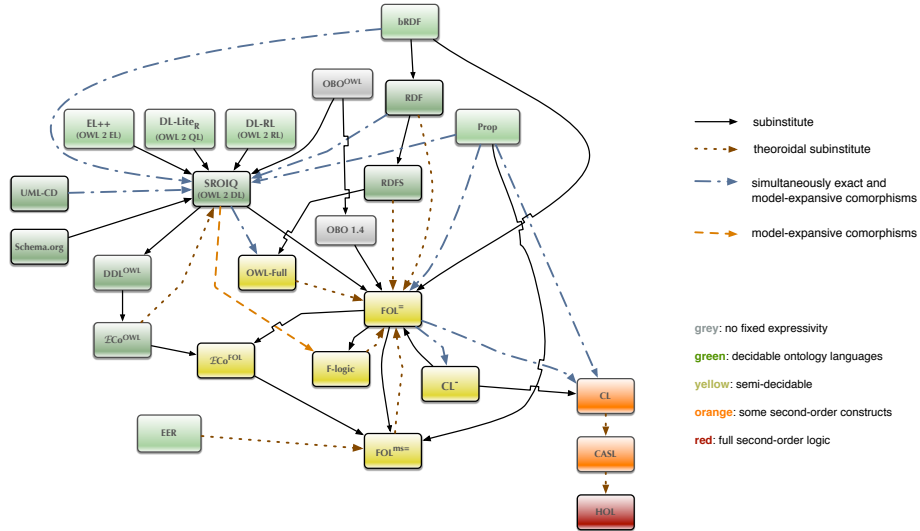


Fig. 5. The logic translation graph for DOL-conforming languages

### 4.3 A Registry for Ontology Languages and Mappings

The OntoIOP standard is not limited to a fixed set of ontology languages. It will be possible to use any (future) logic or mapping (in the sense of Sect. 4) with DOL. This led to the idea of setting up a *registry* to which the community can contribute descriptions of any logics and mappings. Moreover, logics can support ontology languages (e.g.  $SROIQ(D)$  [8] supports OWL), which can in turn have different serializations. All these notions are part of the LoLa ontology. LoLa turns Ontohub itself into part of the Semantic Web: it is mostly written in RDF (the data part) and OWL (the concepts), but also contains first-order parts. We use RDF and OWL reasoners in order to derive new facts in LoLa. A full description and discussion of the LoLa ontology can be found in [13].

Fig. 6 shows the top-level classes of LoLa’s OWL module, axiomatising logics, languages, and mappings to the extent possible in OWL. Object-level classes (that is, classes providing the vocabulary for expressing distributed ontologies) comprise ontologies, their constituents (namely entities, such as classes and object properties, and sentences, such as class subsumptions), as well as links between ontologies. Mappings are modelled by a hierarchy of properties cor-

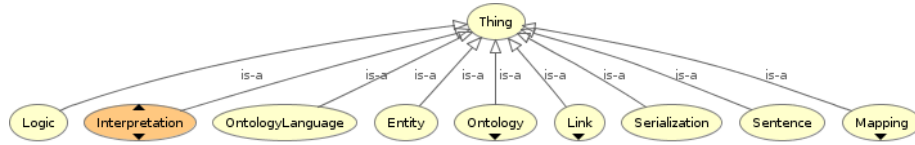


Fig. 6. Top-level classes in the OWL ontology

responding to the different types of edges in Fig. 5; see also Fig. 4. The full LoLa ontology is available at <http://purl.net/dol/1.0/rdf#>.

## 5 Heterogeneous DOL ontologies

Many (domain) ontologies are written in DLs such as *SRTOQ* and its profiles. These logics are characterised by having a rather fine-tuned expressivity, exhibiting (still) decidable satisfiability problems, whilst being amenable to highly optimised implementations.

However, expressivity beyond standard DLs is required for many foundational ontologies (as well as bio-medical ontologies), including DOLCE, BFO<sup>8</sup> and GFO. Moreover, for practical purposes, these foundational ontologies also come in different versions ranging in expressivity, typically between OWL (e.g. DOLCE Light, BFO-OWL) and first-order (DOLCE, GFO) or even second-order logic (BFO-Isabelle).

The relation between such different versions, OWL and first-order, may be recorded in various ways. In some cases it is primarily discussed in the research literature, see the mereo-topological ontology of Keet [10] for an example, or it is described in the OWL ontology within a comment, however not carrying formal semantics. In the latter case, the comment might only contain an *informal explanation* of how the OWL approximation was obtained (DOLCE Light would be an example), but it might also describe a fully formal, axiomatised first-order extension of the OWL ontology. We here briefly describe this last scenario taking the example of BFO-OWL, and show how this information can be faithfully re-written into a heterogeneous DOL ontology with formal semantics.

Consider the object property ‘temporalPartOf’ found in BFO-OWL. The OWL axiomatisation states this to be a transitive subproperty of ‘occurrentPartOf’, and the inverse of ‘hasTemporalPart’.<sup>9</sup> This property is however annotated in a rich way, containing example usages, a richer first-order axiomatisation of this property with pointers to the corresponding axioms in the first-order version, as well as natural language rephrases of these axioms. The DOL ontology below captures the logical part of this annotation as follows: the specification ‘BFO-OWL’ first lists the entire OWL axiomatisation of the ontology. In a second

<sup>8</sup> See <http://www.ifomis.org/bfo/>

<sup>9</sup> Indeed, ‘parthood’ being typically understood as an anti-symmetric relation in mereology is the canonical example of a relation that cannot be adequately formalised in OWL, and a corresponding comment can be found in many bio-medical ontologies.

step, the specification ‘BFOWithAssociatedAxioms’ *imports* BFO-OWL along a translation to Common Logic, and subsequently extends the resulting first-order version of BFO-OWL with the first-order axioms previously only listed as comments. As a result, we obtain a two-level specification of BFO, the original OWL part (being supported by OWL reasoners) and the full first-order part in CLIF Common Logic syntax (amenable to first-order theorem proving and non-conservatively extending the OWL consequences).

```

logic OWL

ontology BFO-OWL =
  ...
  ObjectProperty: temporalPartOf Transitive
  SubPropertyOf: occurrentPartOf
  InverseOf: hasTemporalPart
  ...
end

logic CommonLogic

ontology BFOWithAssociatedAxioms =
  BFO-OWL with OWL2CommonLogic then
  ...
  (forall (x y) (if (properTemporalPartOf x y)
    (exists (z) (and (properTemporalPartOf z y)
      (not (exists (w)
        (and (temporalPartOf w x) (temporalPartOf w z)
          ))))))))

  (iff (properTemporalPartOf a b) (and (temporalPartOf a b) (not (= a b))))

  (iff (temporalPartOf a b) (and (occurrentPartOf a b)
    (exists (t) (and (TemporalRegion t) (occupiesSpatioTemporalRegion a t)))
    (forall (c t1) (if (and (Occurrent c) (occupiesSpatioTemporalRegion c t1)
      (occurrentPartOf t1 r))
      (iff (occurrentPartOf c a) (occurrentPartOf c b))))))
  ...

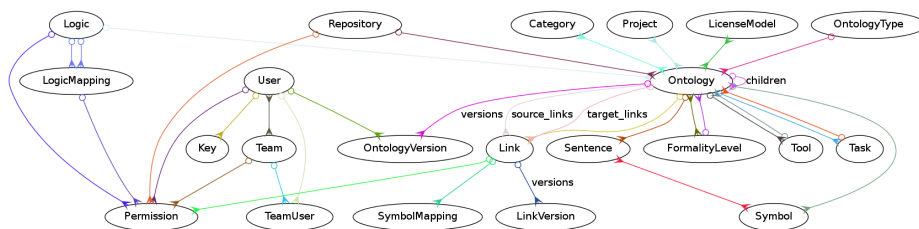
```

Note, however, that the extension to a first-order version is not always as straightforward as in the example just described. The first-order axioms found in the annotation of the property ‘occurrentPartOf’ contain both a binary relation ‘occurrentPartOf’ as well as a ternary, temporalised relation ‘occurrentPartOf’ (this is allowed in the Wild West syntax of CLIF). Whilst this also can be easily turned into a two-level DOL specification, what is typically missing is *bridging axioms* formalising the formal relationship between the temporalised and non-temporalised version of the relation. However, adding such bridging axioms and establishing formal interpretations between OWL and first-order versions of an ontology is precisely a feature and the strength of the DOL language.

## 6 Architecture of Ontohub

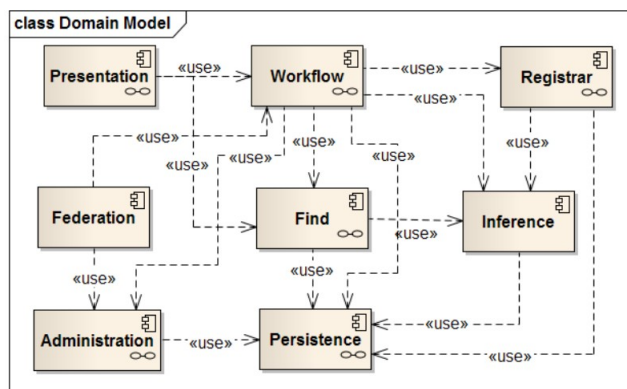
The Ontohub front-end providing the web interface is implemented in Ruby on Rails. Efficient indexing and searching is done via an interface to Apache Lucene – currently Tomcat/Solr, in the future elasticsearch. The database backend is PostgreSQL, but in principle any database supported by Rails (e.g. MySQL, SQLite) could be used. The parsing and inference backend is the Heterogeneous Tool Set (Hets [19], available at <http://hets.eu>). Hets supports a large number of basic ontology languages and logics, and is capable of describing the structural outline of an ontology from the perspective of DOL, which is not committed to one particular logic (see Sect. 3).

A simplified version of the Ontohub database schema is shown in Fig. 7. On the left, one has logics and logic mappings (with source and target logic) — each ontology has a logic. Users, teams, permissions and keys (which are ssh keys regulating the git access to Ontohub) follow, and lead to repositories. Each ontology belongs to a repository, while a repository may comprise many ontologies. This many-to-one relationship is illustrated with circles and arrows at the tips of the link. Ontologies can have different versions, and each version has been created by a specific user and is tied to commits in the git version control system. On the right hand side, you see the different ontology metadata, like categories, formality levels, tools, tasks etc. On the right at the bottom, there are the basic ingredients of ontologies: sentences and symbols. The middle of the bottom shows links between ontologies with their versions. Each link provides a mapping of the symbols in the respective ontologies.



**Fig. 7.** Subset of the Ontohub database schema

In the long run, the architecture of Ontohub will follow that of the Open Ontology Repository (OOR) initiative. This initiative 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 in registry-repositories, where an ontology repository is a facility where ontologies and related information artifacts can be stored, retrieved and managed” [21]. OOR aims at supporting multiple ontology languages, including OWL and Common Logic. OOR is a long-term initiative, which has not



**Fig. 8.** Architecture of the Open Ontology Repository (OOR)

resulted in a complete implementation so far<sup>10</sup>, but established requirements and designed an architecture, see Fig. 8.<sup>11</sup>

The key feature of the OOR architecture 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. Fig. 9 depicts the new Ontohub architecture, which will be realized as a set of decoupled RESTful services<sup>12</sup>, while Ontohub is still at the center of the architecture.

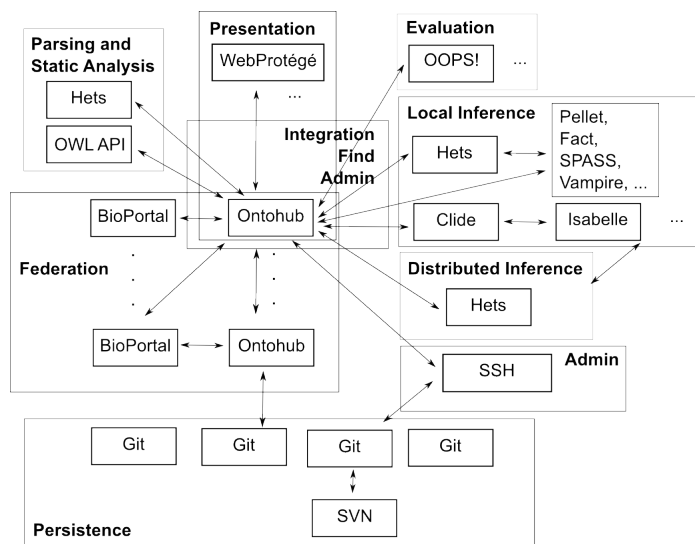
A *federation* API allows the data exchange with among Ontohub and also BioPortal instances. We therefore have generalised the OWL-based BioPortal API to arbitrary ontology languages, e.g. by abstracting classes and object properties to symbols of various kinds. *Parsing and static analysis* is a service of its own, returning the symbols and sentences of an ontology in XML format. Hets can do this for a large variety of ontology languages, while the OWL API does scale better for very large OWL ontologies. That is, some enhanced services may be provided for a restricted set of ontology languages. This is also the case for *presentation*: while Ontohub has a language-independent presentation, WebProtégé provides an enhanced presentation for OWL ontologies. We plan to add enhanced presentation layers for other languages as well (e.g. following the Sigma/SUMO environment for first-order logic). We have already integrated OOPS! [?] 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.

*Local inference* is done by encapsulating standard batch-processing reasoners (Pellet, Fact, SPASS, Vampire etc.) into a RESTful API, as well as through

<sup>10</sup> The main implementation used by OOR is BioPortal, which however does not follow the OOR principles very much.

<sup>11</sup> See <http://tinyurl.com/OOR-Requirement> and <http://tinyurl.com/OOR-Candidate3>, respectively

<sup>12</sup> See <http://tinyurl.com/onto-arch> for detailed API specifications, however not linked to the LoLa ontology yet. OOR already provides an ontologically enriched API.



**Fig. 9.** Ontohub in a network of web services

Hets (which has been interfaced with 15 different reasoners). The integration of interactive provers bears many challenges; a first step is the integration of Isabelle via the web interface Clide [14] developed by colleagues in Bremen, which is currently equipped with an API for this purpose. *Distributed inference* is done via Hets. For example, if an interpretation between two ontologies shall be proved, Hets computes what this means in terms of local inferences, and propagates suitable proof obligations to individual ontologies.

Finally, the *persistence* layer is based on Git (via git-svn, also Subversion repositories can be used). Git provides version control and branching of versions. We have equipped Git with a web interface<sup>13</sup>, such that ontology versions can be directly edited and committed. Moreover, users can also use a Git repository on their local machine, and commits will be immediately available in Ontohub.

## 7 Conclusion and Future Work

Ontohub will be the basis of several coordinated efforts: we intend to set up an instance SpacePortal.org for ontologies in the spatial domain, and Concept-Portal.org for concept blending (see <http://www.coinvent-project.eu>). We also will use federation with BioPortal to integrate biomedical ontologies into Ontohub. Then, with BioPortal’s rich collection of alignments, Ontohub’s ontology combination feature can be systematically used and evaluated. The FOIS 2014 ontology competition has used Ontohub as platform for uploading ontologies used in submissions, see <https://ontohub.org/fois-ontology-competition>. Ontologies used in FOIS papers often need expressiveness beyond OWL; here, the multi-logic nature of Ontohub is essential.

Ontohub plays a double role: it is a repository for ontologies *and* for ontology languages, their underlying logics, and their translations. Currently, the

<sup>13</sup> See <https://github.com/eugenk/bringit>

logics supported by Ontohub are those supported by a corresponding Haskell implementation in the Hets. In the future, we plan to use a logical framework for the purely declarative specification of both logics and translations [4], easing the integration of new logics into Ontohub and simultaneously providing a formal reference and a machine-processable description, thus deepening the role of Ontohub not only being *about* the Semantic Web, but also *part* of it.

## Acknowledgements

The development of Ontohub and DOL is supported by the German Research Foundation (DFG), Project I1-[OntoSpace] of the SFB/TR 8 “Spatial Cognition”. The project COINVENT acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant number: 611553.

The authors would moreover like to thank the OntoIOp working group for their valuable input, particularly Michael Grüninger, Maria Keet, Christoph Lange, Fabian Neuhaus and Peter Yim. We also thank the OOR community and the Ontology Summit 2013 Hackathon participants, especially Ken Baclawski, María Poveda Villalon and Peter Yim. Last but not least, we thank Hardik Balar, Ingo Becker, Christian Clausen, Daniel Couto Vale, Sascha Graef, Timo Kohorst, Julian Kornberger, Eugen Kuksa, Christian Maeder, Henning Müller, Tim Reddehase and Sören Schulze for doing the implementation work.

## References

1. The Tones repository. <http://www.inf.unibz.it/tones>.
2. *The NeOn Ontology Engineering Toolkit*, 2008. <http://www.neon-project.org/>.
3. Tomasz Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286:197–245, 2002.
4. M. Codescu, T. Mossakowski, and O. Kutz. A categorical approach to ontology alignment. In *Proc. of the 9th International Workshop on Ontology Matching (OM-2014)*. CEUR-WS, 2014. To appear.
5. Mihai Codescu, Fulya Horozal, Michael Kohlhase, Till Mossakowski, and Florian Rabe. Project abstract: Logic atlas and integrator (LATIN). In James H. Davenport, William M. Farmer, Josef Urban, and Florian Rabe, editors, *Intelligent Computer Mathematics 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011, Bertinoro, Italy, July 18-23, 2011. Proceedings*, volume 6824 of *Lecture Notes in Computer Science*, pages 289–291. Springer-Verlag Berlin Heidelberg, 2011.
6. J. Goguen and G. Rosu. Institution morphisms. *Formal aspects of computing*, 13:274–307, 2002.
7. J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39:95–146, 1992. Predecessor in: LNCS 164, 221–256, 1984.
8. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible *SRCIQ*. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 57–67. AAAI Press, June 2006.

9. Information technology – Common Logic (CL): a framework for a family of logic-based languages, 2007. ISO/IEC 24707:2007.
10. C Maria Keet and Alessandro Artale. Representing and reasoning over a taxonomy of part–whole relations. *Applied Ontology*, 3(1):91–110, 2008.
11. Michael Kifer and Georg Lausen. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. In *ACM SIGMOD Record*, volume 18, pages 134–146. ACM, 1989.
12. O. Kutz, T. Mossakowski, and D. Lücke. Carnap, Goguen, and the Hyperontologies: Logical Pluralism and Heterogeneous Structuring in Ontology Design. *Logica Universalis*, 4(2), 2010. Special issue on ‘Is Logic Universal?’.
13. Christoph Lange, Till Mossakowski, and Oliver Kutz. LoLa: A Modular Ontology of Logics, Languages, and Translations. In Thomas Schneider and Dirk Walther, editors, *Workshop on modular ontologies*, volume 875 of *CEUR-WS online proceedings*, 2012.
14. Christoph Lüth and Martin Ring. A web interface for Isabelle: The next generation. In *Intelligent Computer Mathematics*, pages 326–329. Springer Berlin Heidelberg, 2013.
15. José Meseguer. General logics. In H. J. Ebbinghaus, editor, *Logic Colloquium ’87*, pages 275–329. North Holland, 1989.
16. Till Mossakowski and Oliver Kutz. The Onto-Logical Translation Graph. In Oliver Kutz and Thomas Schneider, editors, *Modular Ontologies*, number 230 in *Frontiers in Artificial Intelligence and Applications*, pages 94–109. IOS Press, September 2011.
17. Till Mossakowski, Oliver Kutz, and Christoph Lange. Semantics of the distributed ontology language: Institutes and institutions. In Narciso Martí-Oliet and Miguel Palomino, editors, *Recent Trends in Algebraic Development Techniques, 21th International Workshop, WADT 2012*, volume 7841 of *Lecture Notes in Computer Science*, pages 212–230. Springer, 2013.
18. Till Mossakowski, Christoph Lange, and Oliver Kutz. Three Semantics for the Core of the Distributed Ontology Language. In Maureen Donnelly and Giancarlo Guizzardi, editors, *7th International Conference on Formal Ontology in Information Systems (FOIS)*, volume 239 of *Frontiers in Artificial Intelligence and Applications*, pages 337–352. IOS Press, 2012. FOIS Best Paper Award.
19. Till Mossakowski, Christian Maeder, and Klaus Lüttich. The Heterogeneous Tool Set. In Orna Grumberg and Michael Huth, editors, *TACAS 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer-Verlag Heidelberg, 2007.
20. Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute, et al. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic acids research*, 37(suppl 2):W170–W173, 2009.
21. Open Ontology Repository (OOR), 2012.
22. María Poveda-Villalón, Mari Carmen Suárez-Figueroa, and Asunción Gómez-Pérez. Validating Ontologies with OOPS! In *Knowledge Engineering and Knowledge Management*, pages 267–281. Springer, 2012.
23. Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.