Ontohub - a repository engine for heterogeneous ontologies and alignments

Till Mossakowski[1,2], Oliver Kutz[1], and Mihai Codescu[3]

[1] Collaborative Research Centre on Spatial Cognition, University of Bremen
[2] DFKI GmbH Bremen
[3] University of Erlangen-Nürnberg

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, as well as ontology alignments.
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 part of the emerging Ontology Integration and Interoperability standard OntoIOp (ISO Working Draft 17347).

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. Moreover, ontology alignment plays a crucial role: alignments can relate ontologies into networks of ontologies, and new ontologies can be created from such networks via module extraction and combination along alignments, cf. [14].

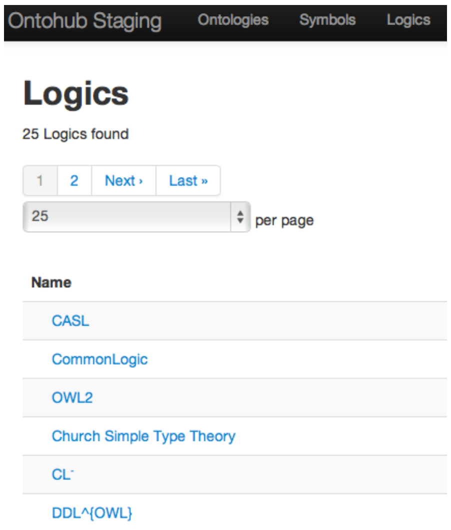
Existing ontology search engines and repositories include search engines like Swoogle, Watson, and Sindice. They concentrate on (full-text and structured) search and querying. TONES [1] is a repository for OWL ontologies that provides some metrics, as well as an OWL sublanguage analysis. BioPortal [23] 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 are supported. NeON [2] is a toolbox for searching, selecting, comparing, transforming, aligning and integrating ontologies. Besides OWL, also F-logic is supported.

Ontohub enjoys the following distinctive features:

– 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 (OWL, RDF, Common Logic, first-order logic, relational database schemes, planned: UML, F-logic, distributed description logics, and more),
– 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 the distributed ontology language (DOL), introduced in the next section. Users of Ontohub can upload, browse, search and annotate basic ontologies in various languages via a web frontend. See Fig. 1, which shows an excerpt of logics available in Ontohub (currently 25), and `http://ontohub.org`. Ontohub is open source, the sources are available at `https://github.com/ontohub/ontohub`.

2 The Distributed Ontology Language (DOL) – Overview



Logics

25 Logics found

Name
CASL
CommonLogic
OWL2
Church Simple Type Theory
CL⁻
DDL^{OWL}

Fig. 1. `ontohub.org` portal: overview of logics

The Distributed Ontology Language (DOL), currently being standardized within the OntoIOp (Ontology Integration and Interoperability) activity[4] of ISO/TC 37/SC 3, 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

[4] For details and earlier publications, see the project page at `http://ontoiop.org`.

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.[5] 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 [21, 20], and the general theory of heterogeneous specifications for ontologies in [13]. DOL uses internationalized resource identifiers (IRIs) for all its entities in order to foster linked data compliance.

2.1 Foundations

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 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 amount to a category of signatures. Readers unfamiliar with category theory may replace this with a partial order (signature morphisms are then just inclusions). See [20] 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* $\mathbf{Sign}$ *of signatures and signature morphisms;*
– *a sentence functor*[6] $\mathbf{Sen} : \mathbf{Sign} \to \mathbb{S}et$ *assigning to each signature the set of its sentences and to each signature morphism* $\sigma : \Sigma \to \Sigma'$ *a sentence translation function* $\mathbf{Sen}(\sigma) : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$*;*
– *a set* $\mathbf{Symbols}$ *of symbols and a set* $\mathbf{Kinds}$ *of symbol kinds together with a function* $\mathbf{kind} : \mathbf{Symbols} \to \mathbf{Kinds}$ *giving the kind of each symbol;*
– *a functor* $\mathbf{Sym} : \mathbf{Sign} \to \mathbb{S}et$ *assigning to each signature* Σ *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* [8]. Similarly, we can complement a logic syntax with a proof theory, introducing a derivability relation between sentences, thus obtaining an *entailment system* [17]. In particular, this can be done for all logics in use in Ontohub.

[5] The languages that we call "basic" ontology languages here are usually limited to one logic and do not provide meta-theoretical constructs.
[6] If running between partial orders, a functor is just a mapping.

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 Σ, 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 (Σ, E) where Σ is a signature and E is a set of Σ-sentences. Given an ontology O, we denote by $\mathsf{Sig}(O)$ the signature of the ontology. An ontology morphism $\sigma : (\Sigma_1, E_1) \to (\Sigma_2, E_2)$ is a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ such that $\sigma(E_1)$ is a logical consequence of E_2. Several notions of *translations* between logics can be

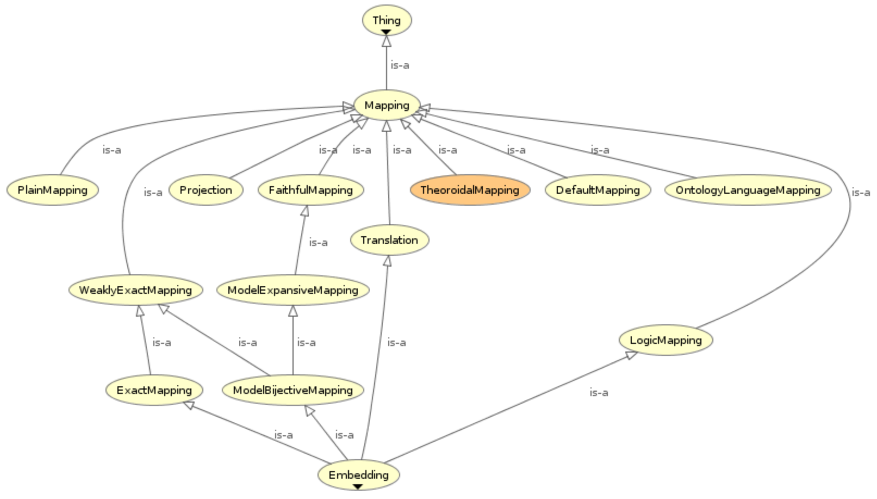


Fig. 2. The part of the OWL ontology concerning mappings

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 Φ and Σ-sentences in L_1 to $\Phi(\Sigma)$-sentences in L_2, for each L_1-signature Σ, in a compatible way with the sentence translations along morphisms. The complexity of translation increases when a model theory or a proof theory is added to the logic syntax. Fig. 2 shows the inferred class hierarchy below the class `Mapping` of the LoLa ontology (see Sect. 2.3 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 [7].

– *plain mapping* versus *simple theoroidal mapping* [7]: 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 [19] 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 [4].

2.2 A Graph of Logic Translations

Fig. 3 is a revised and extended version of the graph of logics and translations introduced in [19]. 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 [21, 19]. 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. 2.3).

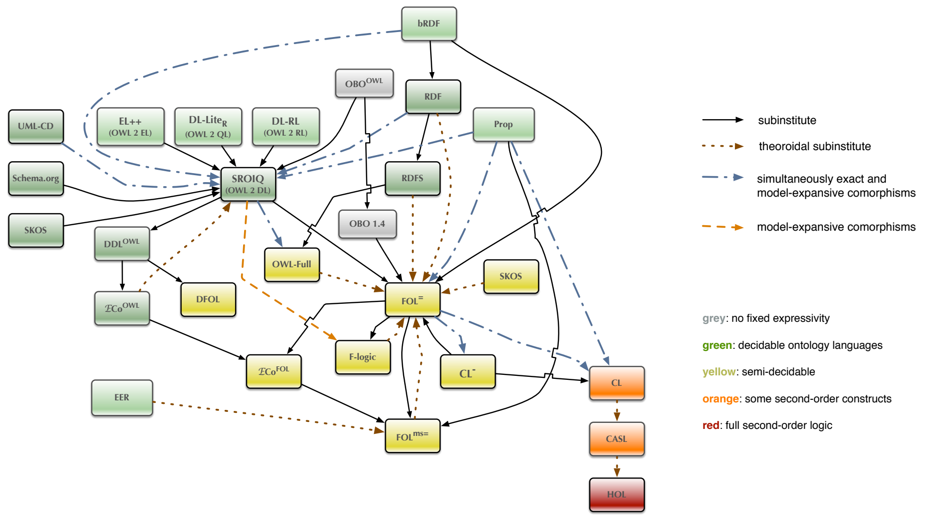


Fig. 3. The logic translation graph for DOL-conforming languages

2.3 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. 2.1) 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. $\mathcal{SROIQ}(D)$ [10] supports OWL), which can in turn have different serializations. All these notions are port 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 [15].

Fig. 4 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 corre-

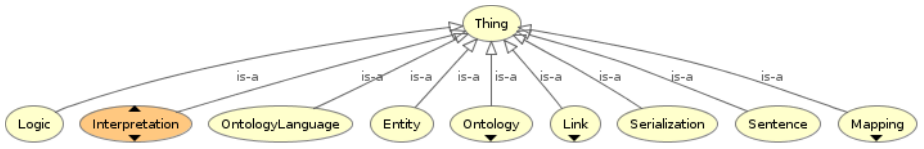


Fig. 4. Top-level classes in the OWL ontology

sponding to the different types of edges in Fig. 3; see also Fig. 2. The full LoLa ontology is available at `http://purl.net/dol/1.0/rdf#`.

3 Heterogeneous DOL ontologies

Many (domain) ontologies are written in DLs such as $\mathcal{SROIQ}$ 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), for instance DOLCE[7], BFO[8], or GFO[9]. 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

[7] See `http://www.loa.istc.cnr.it/DOLCE.html`
[8] See `http://www.ifomis.org/bfo/`
[9] See `http://www.onto-med.de/ontologies/gfo/`

literature, see the mereo-topological ontology of Keet [11] 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 'occurent-PartOf', and the inverse of 'hasTemporalPart'.[10] 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 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).

```
%prefix( :     <http://www.example.org/BFO#>
         owl   <http://www.w3.org/2002/07/owl#>
         log   <http://purl.net/dol/logic/> %% descriptions of logics ...
         trans <http://purl.net/dol/translations/> )% %% ... and translations
logic OWL

spec BFO-OWL =
  ...
  ObjectProperty: temporalPartOf Transitive
     SubPropertyOf: occurentPartOf
     InverseOf: hasTemporalPart
  ...
end

logic CommonLogic

spec BFOWithAssociatedAxioms =
   BFO-OWL with OWL2CommonLogic then
  ...
   (forall (x y) (if (properTemporalPartOf x y)
                     (exists (z) (and (properTemporalPartOf z y)
```

[10] 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.

```
                     (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 'occurentPartOf' contain both a binary relation 'occurentPartOf' as well as a ternary, temporalised relation 'occurentPartOf' (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.

4 Architecture of Ontohub

The current architecture of Ontohub is shown in Fig. 5. The front-end providing the web interface is implemented in Ruby on Rails. Tomcat/Solr is used for efficient indexing and searching. The database backend is PostgreSQL, but any database supported by Rails could be used. The parsing and inference backend

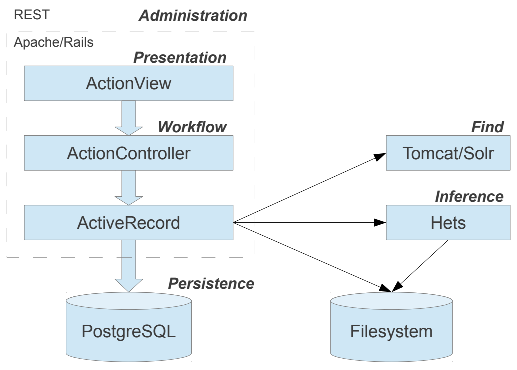


Fig. 5. Current architecture of Ontohub

is the Heterogeneous Tool Set (Hets [18, 22]). 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. 2). This structural information is stored in the Ontohub database and exposed to human users via a web interface and to machine clients as RDF linked data [9]. Beyond basic ontologies, Ontohub supports linking ontologies, across ontology languages, and creating distributed ontologies as sets of basic ontologies and links among them, as can be seen from the left half of the diagram in Fig. 6, which closely corresponds to the abstract syntax of DOL. Note that the Ontohub database schema takes advantage of another

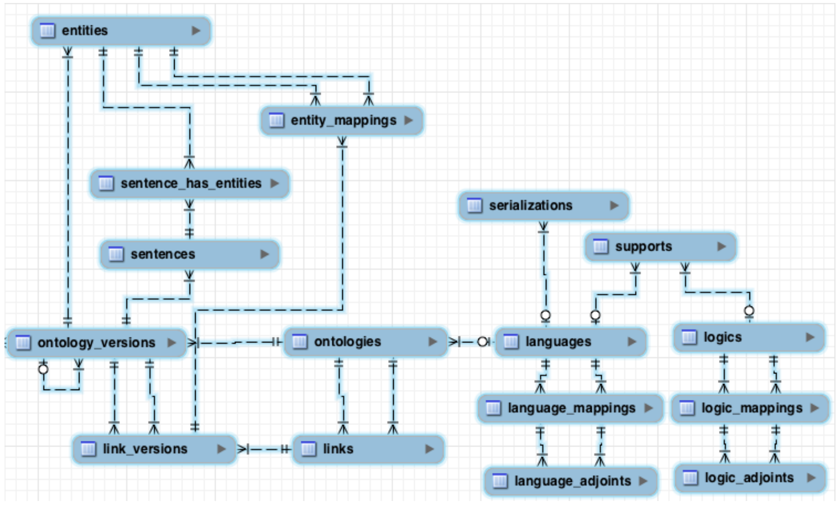


Fig. 6. Subset of the Ontohub database schema (entity-relationship diagram using crow's foot notation); left side: ontologies; right side: OntoIOp registry (cf. section 2.3)

useful abstraction: Same as basic ontologies, we treat distributed ontologies as ontologies. The entities of distributed ontologies are ontologies (basic, or, in complex scenarios, again distributed), and their sentences are links.

The Open Ontology Repository (OOR) 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" [24]. OOR aims at supporting multiple ontology languages, including OWL and Common Logic. OOR is a long-term initiative, which has not resulted in a complete implementation so far[11], but established requirements and designed an architecture, see Fig. 7.[12]

The key feature of the OOR architecture is the decoupling into decentralized services, which are ontologically described (thus arriving at Semantic Web services). With Ontohub, we are moving towards this architecture, while keeping a

[11] The main implementation used by OOR is BioPortal, which however does not follow the OOR principles very much.

[12] See `http://tinyurl.com/OOR-Requirement` and `http://tinyurl.com/OOR-Candidate3`, respectively

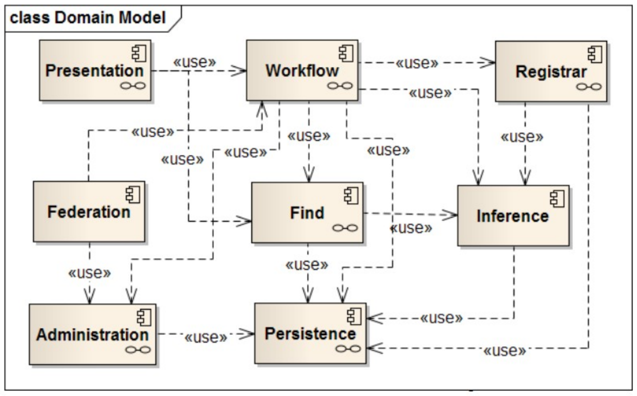


Fig. 7. Architecture of the Open Ontology Repository (OOR)

running and usable system. Fig. 8 depicts the new Ontohub architecture, which will be realized as a set of decoupled RESTful services[13], 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 generalized 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 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 generalized 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 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 [16] developed by colleagues in Bremen, which 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.

[13] 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.

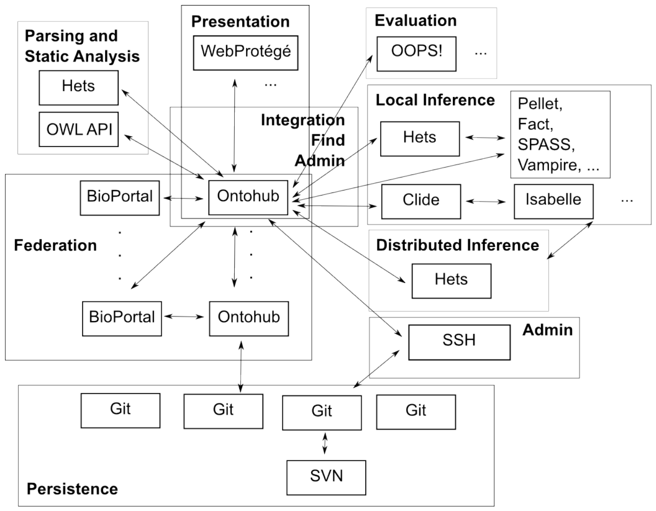


Fig. 8. Ontohub in a network of web services

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[14], 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.

5 Alignment as Colimit Computation

Ontology matching and alignment is a key mechanism for linking the diverse datasets and ontologies arising in the Semantic Web. The management of ontology alignments and especially their combination is a key novel feature of Ontohub, hence we will delve into this topic in some more detail in this section. Ontology matching and alignment based on statistical methods is a relatively developed field, with yearly competitions since 2004 comparing the various strengths and weaknesses of existing algorithms.[15]

An essential part of the matching and alignment process is to relate and identify signature elements from different ontologies (possibly formulated in different ontology languages). In [12] we introduced a general approach for representing alignments, based on the category-theoretical notions of diagram, colimit and pushout. This uniformly captures various 'shapes' of alignments previously introduced in literature. The limitation of the approach in [12] is that the diagram of ontologies to be aligned is defined only implicitly. It is remedied here.

The general alignment format introduced by the Alignment API [6] is represented in the DOL language using the following syntax:

[14] See https://github.com/eugenk/bringit
[15] See http://oaei.ontologymatching.org/ and the overview [25].

alignment A $t_1 : t_2$: O_1 **to** O_2 =
  s_1^1 REL^1 s_2^1,
  ...,
  s_1^n REL^n s_2^n
end

where O_1 and O_2 are the ontologies to be aligned, $t_1 : t_2$ gives the type or the arity of the alignment mapping and its converse, with possible values '1' for injective and total, '+' for total, '?' for injective and '*' for none, s_1^i and s_2^i are O_1 and respectively O_2 symbols, for $i = 1, \ldots, n$, and s_1^i REL^i s_2^i is a *correspondence* which identifies a relation between the ontology entities, either using a relation name or a symbol: $>$ (subsumes), $<$ (is subsumed), $=$ (equivalent), % (incompatible), $\in$ (instance) or $\ni$ (has instance).[16]

The semantics of alignments is given by the following definition.

Definition 2. *Let A be an alignment (using the notations above). The diagram of the alignment is of the following shape (note that the shape is a combined V- and W-alignment in the sense of [26]):*

$$\begin{tikzcd}
O_1 & & B & & O_2 \\
& O'_1 \arrow[ul] \arrow[ur] & & O'_2 \arrow[ul] \arrow[ur] & \\
& & O_1\_O_2 \arrow[uull, bend left, "\sigma_1"] \arrow[uurr, bend right, "\sigma_2"'] & &
\end{tikzcd}$$

where $O_1\_O_2$ contains, for each correspondence $s_1 = s_2$ of A, a symbol $s_1\_s_2$ with the same kind as s_1 and s_2, O'_1 and O'_2 contain the symbols of O_1 and O_2, respectively, which appear in A in a correspondence s_1 REL s_2 such that REL is not equivalence and B is an ontology constructed (in a logic-dependent manner, according to the interpretation of the specified relations in the underlying logic) from the relations between the elements of O'_1 and O'_2 specified in the alignment. The signature morphisms σ_1 and σ_2 map each symbol $s_1\_s_2$ to s_1 and respectively s_2.

The alignment is ill-formed when it contains an equivalence between symbols of different kinds, if B fails to be a correct ontology, or if the alignment mapping does not have its specified arities.

Note that in some cases it is possible to optimize the shape of the diagram to a simpler one: e.g. if the alignment mapping is functional and all the correspondences are equivalences, we obtain an ontology interpretation between O_1 and O_2.

DOL also provides means for combining a diagram of ontologies into a new ontology, such that the symbols related in the diagram are identified. The syntax of combinations is

[16] In the alignment API and in DOL, one can also express confidence values for correspondences. We here assume that they are all 1.

combine $l_1 : O_1, \ldots, l_m : O_m, M_1, \ldots, M_n, A_1, \ldots, A_p$ `excluding` $M'_1, \ldots, M'_k$

where l_i are labels, O_i are ontologies, M_i, M'_j are morphisms and A_i are alignments. The semantics of such a combination is given in terms of a *colimit*. We refrain from presenting the category-theoretic definition here (which can be found in [3]). A *diagram* consists of a set ontologies and a set of ontology morphisms between them. The *colimit* of a diagram is similar to a disjoint union of its ontologies, with some identifications of shared parts as specified by the morphisms in the diagram.

The user specifies a diagram D formed with the ontologies O_i, the morphisms M_i and the alignments A_i. Together with two ontologies O_1 and O_2, any path of imports between O_1 and O_2 is included in the diagram of the combination. The user may decide to omit some of the morphisms using the excluding list. The semantics of the combination is then defined as the colimit of the diagram. The role of the labels l_i is to distinguish between identical symbols with different meanings: if a symbol s appears in the ontologies O_1 with label l_1 and O_2 with label l_2, respectively, but there is no common origin for the two occurrences in the diagram, we will obtain in the colimit the symbols $l_1 : s$ and $l_2 : s$.

Example 2.

```
%prefix( :     <http://www.example.org/alignment#>
         owl   <http://www.w3.org/2002/07/owl#>
         log   <http://purl.net/dol/logic/> %% descriptions of logics ...
         trans <http://purl.net/dol/translations/> )% %% ... and translations

distributed-ontology Alignments

language lang:OWL2 logic log:SROIQ syntax ser:OWL2/Manchester

alignment Alignment1 : { Class: Woman } to { Class: Person } =
  Woman < Person
end

ontology AlignedOntology1 =
  combine Alignment1
end

ontology Onto1 =
  Class: Person
  Class: Woman SubClassOf: Person
  Class: Bank
end

ontology Onto2 =
  Class: HumanBeing
  Class: Woman SubClassOf: HumanBeing
  Class: Bank
end

alignment VAlignment : Onto1 to Onto2 =
  Person = HumanBeing,
  Woman = Woman
end

ontology VAlignedOntology =
  combine 1 : Onto1, 2 : Onto2, VAlignment
  %% 1:Person is identified with 2:HumanBeing
  %% 1:Woman is identified with 2:Woman
  %% 1:Bank and 2:Bank are kept distinct
end

ontology VAlignedOntologyRenamed =
  VAlignedOntology with 1:Bank |-> RiverBank, 2:Bank |-> FinancialBank,
                      Person_HumanBeing |-> Person
end
```

The diagram of the first alignment is

$$\begin{tikzcd}
\{Woman\} & & Woman \sqsubseteq Person & & \{Person\} \\
& \{Woman\} \arrow[ul] \arrow[ur] & & \{Person\} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

The diagram of the second alignment is

$$\begin{tikzcd}
Onto1 & & Onto2 \\
& \{Woman, Person\_HumanBeing\} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

The diagram of VAlignedOntology is

$$\begin{tikzcd}
& \{Woman, Person\_HumanBeing, 1:Bank, 2:Bank\} & \\
Onto1 \arrow[ur, dotted] & & Onto2 \arrow[ul, dotted] \\
& \{Woman, Person\_HumanBeing\} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

where the dotted arrows are automatically computed via the colimit.

6 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 ConceptPortal.org for concept blending. 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 conference will use Ontohub as platform for uploading ontologies used in submissions; there will also be a 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 logics supported by Ontohub are those supported by a corresponding Haskell implementation in the Heterogeneous Tool Set (Hets). In the future, we plan to use a logical framework for the purely declarative specification of both logics and translations [5], 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 DOL is supported by the German Research Foundation (DFG), Project I1-[OntoSpace] of the SFB/TR 8 "Spatial Cognition". The authors would like to thank the OntoIOp working group within ISO/TC 37/SC 3 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 Ingo Becker, Christian Clausen, Daniel Couto Vale, Timo Kohorst, Julian Kornberger, Eugen Kuksa, Christian Maeder, Henning Müller 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. Adámek, J., Herrlich, H., and Strecker, G. *Abstract and Concrete Categories.* Wiley, New York, 1990.
4. Borzyszkowski, T. Logical systems for structured specifications. *Theoretical Computer Science 286* (2002), 197–245.
5. Codescu, M., Horozal, F., Kohlhase, M., Mossakowski, T., and Rabe, F. Project abstract: Logic atlas and integrator (LATIN). In *Intelligent Computer Mathematics 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011, Bertinoro, Italy, July 18-23, 2011. Proceedings* (2011), J. H. Davenport, W. M. Farmer, J. Urban, and F. Rabe, Eds., vol. 6824 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, pp. 289–291.
6. David, J., Euzenat, J., Scharffe, F., and dos Santos, C. T. The alignment API 4.0. *Semantic Web 2*, 1 (2011), 3–10.
7. Goguen, J., and Roşu, G. Institution morphisms. *Formal aspects of computing 13* (2002), 274–307.
8. Goguen, J. A., and Burstall, R. M. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery 39* (1992), 95–146. Predecessor in: LNCS 164, 221–256, 1984.
9. Heath, T., and Bizer, C. *Linked Data: Evolving the Web into a Global Data Space*, 1 ed. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, San Rafael, CA, 2011.

10. Horrocks, I., Kutz, O., and Sattler, U. The Even More Irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)* (June 2006), AAAI Press, pp. 57–67.
11. Keet, C. M., and Artale, A. Representing and reasoning over a taxonomy of part–whole relations. *Applied Ontology 3*, 1 (2008), 91–110.
12. Kutz, O., Mossakowski, T., and Codescu, M. Shapes of Alignments - Construction, Combination, and Computation. In *International Workshop on Ontologies: Reasoning and Modularity (WORM-08)* (2008), U. Sattler and A. Tamilin, Eds., vol. 348 of *CEUR-WS online proceedings*.
13. Kutz, O., Mossakowski, T., and Lücke, D. Carnap, Goguen, and the Hyperontologies: Logical Pluralism and Heterogeneous Structuring in Ontology Design. *Logica Universalis 4*, 2 (2010). Special issue on 'Is Logic Universal?'.
14. Kutz, O., Normann, I., Mossakowski, T., and Walther, D. Chinese Whispers and Connected Alignments. In *Proc. of the 5th International Workshop on Ontology Matching (OM-2010)* (9th International Semantic Web Conference ISWC-2010, November 7, 2010, Shanghai, China., 2010).
15. Lange, C., Mossakowski, T., and Kutz, O. LoLa: A Modular Ontology of Logics, Languages, and Translations. In *Workshop on modular ontologies* (2012), T. Schneider and D. Walther, Eds., vol. 875 of *CEUR-WS online proceedings*.
16. Lüth, C., and Ring, M. A web interface for isabelle: The next generation. In *Conferences on Intelligent Computer Mathematics CICM 2013*, J. Carette, J. H. Davenport, W. Windsteiger, P. Sojka, D. Aspinall, and C. Lange, Eds., Springer LNCS. to appear.
17. Meseguer, J. General logics. In *Logic Colloquium '87*, H. J. Ebbinghaus, Ed. North Holland, 1989, pp. 275–329.
18. Mossakowski, T. Hets: the Heterogeneous Tool Set. http://hets.dfki.de.
19. Mossakowski, T., and Kutz, O. The Onto-Logical Translation Graph. In *Modular Ontologies* (2011), O. Kutz and T. Schneider, Eds., IOS.
20. Mossakowski, T., Kutz, O., and Lange, C. Semantics of the distributed ontology language: Institutes and institutions. In *Recent Trends in Algebraic Development Techniques, 21th International Workshop, WADT 2012* (2013), N. Martí-Oliet and M. Palomino, Eds., vol. 7841 of *Lecture Notes in Computer Science*, Springer, pp. 212–230.
21. Mossakowski, T., Lange, C., and Kutz, O. Three Semantics for the Core of the Distributed Ontology Language. In *7th International Conference on Formal Ontology in Information Systems (FOIS)* (2012), M. Donnelly and G. Guizzardi, Eds., vol. 239 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 337–352. FOIS Best Paper Award.
22. Mossakowski, T., Maeder, C., and Lüttich, K. The Heterogeneous Tool Set. In *TACAS 2007* (2007), O. Grumberg and M. Huth, Eds., vol. 4424 of *Lecture Notes in Computer Science*, Springer-Verlag Heidelberg, pp. 519–522.
23. Noy, N. F., Shah, N. H., Patricia L. Whetzel, ., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C. G., and Musen, M. A. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research 37* (2009), W170–W173. `http://bioportal.bioontology.org`.
24. Open Ontology Repository (OOR), 2012.
25. Pavel, S., and Euzenat, J. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering 99* (2012).
26. Zimmermann, A., Krötzsch, M., Euzenat, J., and Hitzler, P. Formalizing Ontology Alignment and its Operations with Category Theory. In *Proc. of FOIS-06* (2006), pp. 277–288.