Ontohub

Version Control, Linked Data and Theorem Proving for Ontologies

Eugen Kuksa [a] Till Mossakowski [a]
[a] *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 as well as various structuring and modularity constructs and inter-theory (concept) mappings, building on the OMG-standardized DOL language.

Ontohub aims at satisfying a subset of the requirements for an Open Ontology Repository (OOR). OOR is a long-term international initiative, which established requirements and designed an architecture. Ontohub is the first repository engine meeting a substantial amount of OOR's requirements, including an API for federation as well as support for logical inference and axiom selection.

Keywords. ontology, repository, Git, linked data, logical inference, heterogeneity

1. Concept and Central Features of Ontohub

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 already in use is steadily growing. Hence, facilities for organizing ontologies, searching and maintenance are becoming more important. Ontohub is a novel web-based repository engine. Its distinctive features are:

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 URL is used for referencing an ontology, downloading it (for use with tools), and for user-friendly presentation in the browser,

modular architecture Ontohub consists of components that communicate via RESTful APIs,

logical inference interfaces to various theorem provers provide reasoning support,

multi-language support for OWL, Common Logic and others.

Concerning the last feature, Ontohub fully supports the Distributed Ontology Language (DOL), which is an Object Management Group (OMG) specification,

see [1,2] and http://dol-omg.org. DOL provides a unified framework for (1) ontologies formalized in heterogeneous logics like OWL, Common Logic, TPTP and higher-order logic, (2) modular ontologies, (3) mappings between ontologies including ontology alignments, interpretation of theories, conservative extensions, translation to other ontology languages – all equipped with a formal semantics.

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 https://github.com/ontohub/ontohub. Currently, Ontohub has 200 registered users, which include ontology researchers, ontology developers as well as master and PhD students.

2. Related Work

Existing ontology resources on the web include search engines like Swoogle, Watson, and Sindice. They concentrate on (full-text and structured) search and querying. Ontology repositories also provide persistent storage and maintenance. TONES [3] is a repository for OWL [4] ontologies that provides some metrics, as well as an OWL sublanguage analysis. BioPortal [5] 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 Open Biomedical Ontologies (OBO) [6] are supported. The NeOn Toolkit [7] supports searching, selecting, comparing, transforming, aligning and integrating ontologies. It is based on the OWL API and is no longer actively maintained.

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" [8]. 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. OOR is a long-term initiative, which has not resulted in a complete implementation so far[1], but established requirements and designed an architecture.[2] Ontohub is unique in following OOR's ambitious goals.

3. Working with Ontohub

Ontohub's main use case is to manage ontologies which are stored in repositories in Ontohub. How to accomplish this basic task is described in the video tutorial at http://wiki.ontohub.org/index.php/Tutorial. The user can upload an ontology file to one of Ontohub's existing repositories, or create a new repository to store ontologies in. After the ontology is saved, it is evaluated by the parsing and inference backend. As soon as this step is finished, the user has access to the

[1] The main implementation used by OOR is (a cosmetically adapted) BioPortal, which however does not follow the OOR principles very much.

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

ontology's details like the underlying logic, its symbols and their kind, mappings from and to other ontologies, child ontologies (in case of a DOL ontology library), its axioms and its proof obligations (called theorems in Ontohub for simplicity).

Ontohub stores ontologies as parts of ontology repositories. Each ontology repository in Ontohub is directly linked to a Git repository. Therefore, Ontohub supports all aspects of version control that Git provides, including support for non-linear distributed ontology development. Different versions of an ontology are available via Git-clients, the Ontohub web application and the Ontohub API.

If the user already has a Git or subversion repository containing ontologies, it can be forked or mirrored by Ontohub. This creates a server-local copy of the existing repository in the web application and allows to analyze the ontologies on the server. In case of forking the repository, the new copy can be changed independently of the original repository. In case of mirroring the repository, the copy is a read-only repository in Ontohub which gets synchronized daily with the original one.

Users can be grouped to teams. Both users and teams can be added as *readers*, *editors* or *owners* to a repository for permission management. Repositories can be public or private with respect to readability or writability. While *owners* are able to manage repository metadata (like e.g. permissions and a description text), *editors* are allowed to modify files inside the repository. Ontologies in public repositories are readable by anyone. Ontologies in private repositories, however, are only readable by *owners*, *editors* and designated *readers* of the repository.

Git repositories can be cloned from Ontohub to the local machine via SSH or, for publicly readable repositories via HTTP. SSH access is given as soon as an SSH-key is associated with the user's Ontohub account.

Modification of files can be achieved in several ways: (1) A file can be uploaded via the web interface and the target path points to an existing file. (2) An existing file can be edited directly in the web interface. In both cases, the web application commits a newer version of the file to the Git repository. (3) Files can be modified in a local clone of the Git repository. The local commits can then be pushed to Ontohub via SSH. In all cases, the old version of the ontology persists in the repository and is accessible. For each commit, Ontohub analyzes the ontology files and creates versioned metadata for the ontologies.

If an ontology contains theorems, a user with write permissions can attempt to automatically prove these proof obligations. The web interface for proving presents a list of available provers for the ontology and allows to select some additional parameters like proving timeout and axioms allowed to use in the proof attempt. Ontohub implements a prover-independent variant of the SInE [9] axiom selection heuristic. Entering parameters for this heuristic or selecting axioms one by one for proving are the choices for selecting axioms. For each selected prover, Ontohub runs asynchronous proof attempts in parallel. As soon as a proof attempt is finished, its details can be seen on the web page of the corresponding theorem.

4. Architecture of Ontohub

Fig. 1 depicts the Ontohub architecture. The most challenging part of Ontohub's implementation is the complex tool integration. 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

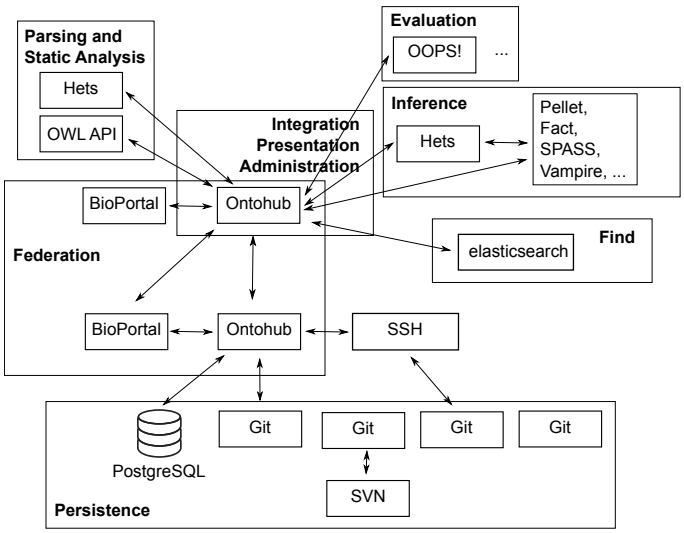

Figure 1. Ontohub in a network of web services

towards the OOR architecture, while keeping a running and usable system. We now briefly describe these 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, but in principle any database supported by Rails (e.g. MySQL, SQLite) could be used. 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 official Git command line client.

Efficient indexing and searching (the *find* layer) is done via elasticsearch.

A *federation* API allows the data exchange among 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.

Parsing and static analysis is a RESTful service of its own provided by the Heterogeneous Tool Set (Hets [10], 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. Hets returns 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. The latter is an example for a service of Ontohub which is provided for a restricted set of ontology languages.

We have integrated OOPS! [11] 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 (Pel-

let, Fact, SPASS, Vampire 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 SInE algorithm is a novelty in this field. In Ontohub, it operates independently of the prover and, thus, supports any prover available in Ontohub.

5. Conclusion and Future Work

Ontohub is on its way from a research prototype to productive use. 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. 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 Isabelle via the web interface Clide [12] developed by colleagues in Bremen, which is currently equipped with an API for this purpose.

References

[1] T. Mossakowski, O. Kutz, M. Codescu, and C. Lange. The Distributed Ontology, Modeling and Specification Language. In C. D. V. et al., editor, *WoMO-13*, volume 1081. CEUR-WS, 2013.

[2] Object Management Group. The distributed ontology, modeling, and specification language (DOL), 2015. OMG draft standard available at `https://ontoiop.org`.

[3] The Tones repository. `http://www.inf.unibz.it/tones`.

[4] I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible $\mathcal{SROIQ}$. In *KR2006*, pages 57–67. AAAI Press, June 2006.

[5] N. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. Storey, C. 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.

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

[7] *The NeOn Ontology Engineering Toolkit*, 2008. `http://www.neon-project.org/`.

[8] Open Ontology Repository (OOR), 2012. `http://oor.net`.

[9] K. Hoder and A. Voronkov. Sine Qua Non for Large Theory Reasoning. In *CADE 23*, pages 299–314, 2011.

[10] T. Mossakowski, C. Maeder, and K. Lüttich. The Heterogeneous Tool Set. In O. Grumberg and M. Huth, editors, *TACAS 2007*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer-Verlag Heidelberg, 2007.

[11] M. Poveda-Villalón, M. C. Suárez-Figueroa, and A. Gómez-Pérez. Validating Ontologies with OOPS! In *Knowledge Engineering and Knowledge Management*, pages 267–281. Springer, 2012.

[12] C. Lüth and M. Ring. A web interface for Isabelle: The next generation. In *Intelligent Computer Mathematics*, pages 326–329. Springer Berlin Heidelberg, 2013.