Software Architectures for Scalable Ontology Networks

Alessandro Adamou[1,2]

[1] Alma Mater Studiorum Università di Bologna, Italy
[2] ISTC, National Research Council, Italy

Abstract. Theory and practice in ontology management are drifting away from monolithic ontologies towards ontology networks. Processing interconnected knowledge models, e.g. through reasoning, can provide greater amounts of explicit knowledge, but at a high computational cost if the whole knowledge base has to be handled by concurrent processes. Our research tackles this problem via a software engineering approach. We devised a framework that combines privileged containers for ontology models with volatile containers for instance data that vary frequently. A reference implementation was developed in an Apache project for content management, and its adoption is providing data and use cases for validating it with objects from Fedora Commons repositories.

1 Introduction

The discipline and methodologies of ontology engineering are gradually shifting away from the monolithic notion of ontologies. Current practices and empirical sciences see reuse as a key criterion for constructing knowledge models today, so that networking-related concepts are being applied to interconnectivity between ontologies. As a consequence, the notion of *ontology network* has begun to surface. An ontology network is created either at design time (e.g. the engineer adds OWL import statements and reuses imported entities) or at a later stage (e.g. someone discovers alignments between multiple ontologies agnostic to each other, creates an ontology with alignment statements and sets up a top ontology that imports all of them). Our research concentrates on exploiting the latter scenario while still accommodating the former.

Establishing ontology networks can have a number of advantages, the most apparent ones being related to redundancy minimisation and reasoning. Interconnecting ontology modules keeps from re-defining basic or shared entities, can augment knowledge on the given entities and produce even more inferred knowledge when reasoners are run on the network. This is, however, when reasoning performance issues kick in. Description Logic (DL) classifies a whole knowledge structure non-selectively, and whether the process is incremental depends on the reasoner implementation. However, if a reasoner is being run for a specific task, portions of this knowledge structure could be unnecessary and inflate the computation to produce results that are no use to the task at hand. Let us consider the social network domain for an example. If the task is to infer a *trust* *network* of users based on their activity, there is hardly any point in including the GeoSpecies ontology that classifies life form species[1], which could however be part of the knowledge base. If, however, another task is to infer an *affinity network*, GeoSpecies could be considered for assessing affinity between zoologists.

There are also considerations to be made on the network substructure. The OWL language relies upon import statements to enforce dependency resolution to a certain extent, and as a de facto standard mechanism it has to be respected. However, import statements are static artifacts that imply strict dependencies identified at design time, therefore the dynamic usage of such OWL constructs has to be delegated to a software platform that serves or aggregates ontologies. We argue it should be combined with an OWL2-compliant versioning scheme.

2 Problem Statement and Research Questions

Given the above challenges, we have formulated these **research questions**:

RQ1. How can a single software framework create ontology networks on top of a common knowledge base, in order to serve them for different processing tasks?

We call these ontology networks *virtual*, in that it is the framework that can create, rewrite and break ontology linkage at runtime, as required by a specific instance-reasoning task, without affecting the logical axioms in the ontologies.

RQ2. Is it possible for such a framework to safely accommodate concurrency in multi-user contexts, and with minimum resource overhead?

Instance data may vary across simultaneous users of a system, but also with time (e.g. only a daily snapshot of data feeds, or the whole history of an ABox for a domain). This variability implies multiple simultaneous virtual networks. A TBox, on the other hand, can be comprised of consolidated schemas and vocabularies, which would be preposterous to copy across ontology networks. However, they are generally the ones with a greater impact on reasoner performance, and users should be able to exclude unneeded parts of them from their networks. We must also make sure that *(i)* changes in a user-restricted ABox do not affect another user's ABox, and *(ii)* the memory footprint of ontology networks in the framework is negligible compared to the combined size of the knowledge base.

RQ3. Can the constructs and limits of standard ontology languages be followed and exploited to this end, without altering the logical structure of an ontology?

Our goal with RQ3 is to make sure that, whatever objects the framework introduces, they can always be exported to legal OWL 2 artifacts; that they do not require yet another annotation schema; that they do not force the addition of logical axioms or the interpretation of existing ones in the original ontologies.

[1] GeoSpecies knowledge base, http://lod.geospecies.org

3 Related Work

Networked architectures. Ontology architecture focuses on applying development and deployment schemes *within* the ontology lifecycle [6]. In this respect, some later efforts in *ontology repository* design such as *Cupboard* [3] are showing some concern over their potential use to implement linked ontology networks, as in the networked model of [5]. The study on ontology architectures has led to the problem of determining the modular decomposition of monolithic ontologies, whose theoretical foundations are extensively described in [7].

Methodologies with reuse support. We are concerned with the importance of reuse in ontology engineering, since reused concepts can be potential interconnection nodes, with little to no alignment effort. The *NeOn Methodololgy* [8] and *eXtreme Design* [2] are examples of such reuse-oriented design methodologies.

Scalable reasoning. Scalability is addressed with regard to reasoning on modularised ontologies [1], but also to adaptively reacting to changes in an ontology. Forward-chaining rule-based reasoners such as LMF[2] have been proposed as a trade-off between expressiveness and computational complexity in large datasets. As for DL reasoning, the approach of Fokoue et al. [4] processes the ontologies at reasoning time through pruning, summarizing and in-memory imaging.

4 Approach and Implementation

The requirement analysis, design, development and evaluation of our proposed solution all occur in the field of content management, with the aim to deliver semantically enhanced capabilities to Content Management Systems (CMS). This setting has allowed us to expand our investigation vertically across knowledge, persistence, interaction and user management, all the while maintaining an angle on industrial adoption trends and software engineering.

Our research and design work opted for a *separation of concerns* between the structures holding class and instance data, following a finer granularity than the classic TBox/ABox pair. We devised an architecture where variable instance data (across time or users) can be orthogonal to vocabularies, meta-models or consolidated instance data. We then distinguished scenarios where some architectural layers should be considered ‘privileged’ for update by applications and CMS administrators, while others (mainly ABox-related) are devoted to volatile instance data supplied by users or external services for single or batch operations. Our framework assembles ontology networks using the following constructs:

- **Session.** A container for instance data loaded at runtime. A user or client application can open a session, load the ontologies containing instance data and attach other parts of the network (see below). For instance, to classify the knowledge extracted from a daily blog post feed, a client can push

[2] Linked Media Framework reasoner,
http://code.google.com/p/kiwi/wiki/Reasoning

the corresponding RDF graph into the same session day-by-day. A session takes ownership of any non-versioned ontologies loaded into it. Although not intended for persistence, it can last across more HTTP sessions over time.
- **Scope.** A "realm" for all the ontologies that model a given domain or context. For instance a "social networks" scope can reference FOAF, SIOC, alignments between these and other related custom models. One or more scopes can be attached to one or more sessions at once, thereby realizing the virtual network model. Each scope is divided in two spaces. The *core* space contains the ontologies that provide immutable knowledge, such as foundational ontologies. The *custom* space extends the core space with additional knowledge such as alignments with controlled vocabularies. Note that one occurrence of the same ontology can be shared across multiple spaces.

When these artifacts are exported as OWL ontologies, linkage relations are materialised as `owl:imports` statements forged for each ontology network, while `owl:versionIRI`s are used by these artifacts to either "claim ownership" of the ontologies they manage or share them (e.g. TBoxes) across networks.

This ontology network management architecture has been implemented as part of the **Apache Stanbol** service platform for semantic content management[3]. Since Stanbol is an extensible framework, any plugin can setup and manage its own ontology scopes and sessions. We contributed the following components to the Stanbol ontology manager package:

- **OntoNet** implements an object model of sessions, scopes and spaces, and comes with Java and RESTful APIs for configuring ontology networks. OntoNet implements policies for determining which ontologies to store persistently and which should be either shared or replicated across networks. The OntoNet constructs are supported by the reasoner features of Stanbol, which implement background jobs and concurrent reasoning services with configurable expressivity.
- **Ontology registry manager** is the facility for referencing ontologies external to Stanbol. By setting up a registry, which is itself an RDF graph, Stanbol can aggregate its ontologies from all over the Web and make them available on-demand or OntoNet to assemble them into ontology networks.

At the time of writing, Stanbol is undergoing its release candidate phase. Our contribution also comes as part of the Interactive Knowledge Stack project[4].

5 Validation

With the reference implementation of our framework in place, we are moving on to the phase of validating it on use cases from the content management domain. Small and medium enterprises have committed to adopt Apache Stanbol. The main use case is provided by a company working in content curation for

[3] Apache Stanbol, http://incubator.apache.org/stanbol
[4] Interactive Knowledge Stack (IKS), http://code.google.com/p/iks-project/

the visual arts domain. There, a Fedora Commons digital object repository[5] is used to maintain image metadata, which are interconnected with a SKOS-based representation of the Getty ULAN repository[6]. OntoNet and reasoners will be employed to produce knowledge enrichments over the standard Fedora metadata vocabulary and present different user interface views on them based on different ontology network configurations. In another use case, OntoNet will be used to manage simultaneous content hierarchies from a shared repository. Scopes are selected depending on the knowledge domains that each user decides to activate, while sessions contain the metadata of user-owned and shared content items.

At the same time, we are developing benchmarking tools to evaluate the computational efficiency of the system *tout-court*. Benchmarking will consider *(i)* the amount and complexity of different networks that can be setup at the same time on the same knowledge base, their memory usage and the minimum Java VM size required; *(ii)* the overhead given by loading the same ontology into multiple scopes or sessions; *(iii)* the duration of DL classification runs on an ontology network large enough to deliver the expected inferences, compared against standard DL reasoners called over the non-pruned knowledge base. The possible size of the knowledge base per se will not be measured as it is strictly bound to the storage mechanism employed. To date, the system effectiveness has been measured through unit-testing and stress tests are guaranteeing that we can set up a scope on a $\sim$ 200 MiB ontology using a VM $\sim$ 1.2 times as large.

References

1. Bao, J.: Representing and reasoning with modular ontologies. Ph.D. thesis, Iowa State University (2007)
2. Blomqvist, E., Presutti, V., Daga, E., Gangemi, A.: Experimenting with eXtreme Design. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 120–134. Springer, Heidelberg (2010)
3. d'Aquin, M., Euzenat, J., Le Duc, C., Lewen, H.: Sharing and reusing aligned ontologies with Cupboard. In: Gil, Y., Noy, N.F. (eds.) K-CAP. ACM (2009)
4. Fokoue, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: The Summary Abox: Cutting Ontologies Down to Size. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 343–356. Springer, Heidelberg (2006)
5. Haase, P., Rudolph, S., Wang, Y., Brockmans, S., Palma, R., Euzenat, J., d'Aquin, M.: D1.1.1 networked ontology model. NeOn Deliverable 1(D1.1.1), 1–60 (2006)
6. Obrst, L.: Ontological Architectures, pp. 27–66. Springer, Netherlands (2010)
7. Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization. LNCS, vol. 5445. Springer, Heidelberg (2009)
8. Suárez de Figueroa Baonza, M.C.: NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse. Ph.D. thesis, Universidad Politécnica de Madrid, Madrid, Spain (2010)

[5] Fedora Commons, http://fedora-commons.org/
[6] Union List of Artist Names,
http://www.getty.edu/research/tools/vocabularies/ulan/