OWL Reasoning in the Real World: Searching for Godot

Kavitha Srinivas

IBM Watson Research Center,P.O.Box 704, Yorktown Heights, NY 10598, USA
ksrinivs@us.ibm.com

Abstract. I will provide an overview of many of the use cases that we looked at to apply OWL ABox reasoning in the real world. The fields we covered included (a) healthcare, and life sciences where the plethora of ontologies might be seen as providing a strong use case for OWL reasoning, (b) information retrieval over unstructured text, (c) master data management, which involves reasoning over product and consumer data incorporated from multiple data sources within an enterprise. In each case, we ran into a series of bottlenecks including incorrect modeling of constructs in the ontology, inherent difficulties in scaling OWL reasoning to real world requirements, needing formalisms outside that of OWL, and needing techniques to semi-automate the construction of ontologies. At least in the use cases we had seen, there is a need for performing TBox OWL reasoning over expressive ontologies, but most realistic uses of ABox reasoning have to be relatively simple in terms of expressivity, for practical reasons.

1 Introduction

In this paper, I will describe SHER (Scalable Highly Expressive Reasoner), a technology we developed to provide semantic querying of large Aboxes using OWL ontologies, and point to lessons learnt from our application of large scale Abox reasoning on realistic use case scenarios.

SHER provides standard description logic reasoning services including consistency checking and conjunctive query answering, and supports the logic OWL-DL excluding nominals and datatypes (i.e., $\mathcal{SHIN}$).

2 SHER System Architecture

SHER relies on a unique combination of an in-memory description logic reasoner and a database backed RDF Store to scale reasoning to very large Aboxes. A key feature of our algorithm is that we perform consistency detection on a summarized version of the Abox rather than the Abox in secondary storage [1]. A summary Abox $\mathcal{A}'$ can be constructed by mapping all individuals in the original Abox $\mathcal{A}$, with the same concept set to a single individual in the summary$\mathcal{A}'$. The summary has three key properties:

(1) for every individual a in the original ABox $\mathcal{A}$, if a has type C in $\mathcal{A}$, the summarized individual a' in $\mathcal{A}'$ also has the same type C
(2) for every pair of individuals (a, b) in the original ABox $\mathcal{A}$, if the relation $R(a, b)$ exists in $\mathcal{A}$, then a relation $\mathrm{R}(a', b')$ holds between their respective summarized individuals in $\mathcal{A}'$.
(3) the same principle as in (2) applies to different-from assertions between pairs of individuals

We have shown that if the summary Abox $\mathcal{A}'$ is consistent w.r.t. a given Tbox $\mathcal{T}$ and a Rbox $\mathcal{R}$, then $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ and $\mathcal{R}$. However, the converse does not hold. In general, an inconsistency in the summary may reflect either a real inconsistency in the original Abox, or could simply be an artifact of the summarization process.

In the case of an inconsistent summary, we use a process of iterative refinement described in [2] to make the summary more precise, to the point where we can conclude that an inconsistent summary $\mathcal{A}'$ reflects a real inconsistency in the actual Abox $\mathcal{A}$. Refinement is a process by which only the part of the summary that gives rise to the inconsistency is made more precise, while preserving the summary Abox properties(1)-(3). To pinpoint the portion of the summary that gives rise to the inconsistency, we focus on the *justification* for the inconsistency, where a justification is a minimal set of assertions which, when taken together, imply a logical contradiction. The refinement process continues until one of two things happen: either the inconsistency disappears (in which the original ABox is deemed consistent); or we are left with an inconsistent portion of a summary that cannot be refined any further (in which case the original ABox is really inconsistent). A key point here is that in either case, we rarely fallback to dealing with specific ABox individuals, and the scalability comes from making decisions on groups of individuals as a whole.

2.1 Scalable Ontology Repository (SOR) subsystem

SOR [3] serves as a high-performance OWL ontology storage system based on Relational Database Management Systems (RDBMS). It is responsible for loading OWL, RDF and triple files into the underlying RDBMS. It also builds and maintains, in the RDBMS, the summary Abox of the whole knowledge base. It can be configured to perform either no inferencing on load or a limited number of inferences[1] on load. Finally, it provides SPARQL and SQL query end-points to the stored knowledge base.

2.2 In Memory Summary Builder (IMSB)

The In Memory Summary Builder (IMSB) module builds, for a given reasoning task, an in-memory version of the relevant subset of the summary Abox stored in the RDBMS. This summary can then be fed to an in-memory reasoner. The

[1] typically application of domain and range inference rules

IMSB module functions in two distinct modes. First, it acts as a filter that filters out irrelevant assertions (from the point of view of the reasoning task at hand) from the summary Abox built in the RDBMS by the SOR module. Second, if a previously built in-memory summary Abox was not precise enough to be conclusive, the IMSB is responsible for building a more precise in-memory summary Abox.

2.3 Fast Sound Reasoner (FSR)

Fast, sound, but not necessarily complete, reasoners (as described in the next section) can be plugged into SHER to quickly find "obvious" answers to a given reasoning task, which can significantly improve the performance of SHER by limiting the number of candidates to test. Section 3 discusses the currently implemented FSRs in SHER.

2.4 Consistency Check and Justification Computation modules

As most DL reasoners, SHER operates by reducing all reasoning tasks to consistency detection [4]. However, unlike other DL reasoners, SHER performs the consistency check on a dramatically reduced in-memory summary Abox. For example, for a membership query $C(x)$, the in-memory filtered summary ABox built by the IMSB module is modified by adding the negation of the query to all summary individuals except those corresponding to "obvious" answers or "obvious" non-answers as determined by a FSR. The modified in-memory summary Abox is then checked for consistency. In the membership query example, a consistent summary indicates that the only solutions are the ones found by the FSR. For an inconsistent summary, a subset of the justifications for the inconsistency is computed. Currently, SHER relies on Pellet [5] for consistency check and justification computation. Note that previous work [6] has shown that the complexity of justification computation for OWL-DL entailments is no worse than the OWL-DL tableau reasoning algorithm, and thus does not affect system performance much.

2.5 Justification Analyzer

Given a justification J for an inconsistency in the in-memory summary Abox, the Justification Analyzer module determines, based on the structure of J [2], whether the portion of the summary corresponding to J needs to be made more precise by the IMSB or whether it is already precise enough to be conclusive. Note that a precise justification denotes a real inconsistency in the ABox, and thus solutions can be derived from information in the justification. On the other hand, an imprecise justification forces the IMSB to perform further refinement of the portion of the summary Abox corresponding to the justification to check for any real inconsistency/solutions. Thus, using justifications makes the entire process efficient and scalable (without it, randomly refining the summary does not make sense).

3 Fast, Sound Reasoners (FSRs)

We have implemented several fast, sound reasoners (FSRs) in SHER that can quickly find a large number of "obvious" solutions to a query. These FSRs can be used independently or can be used in conjunction with SHER to perform sound and complete reasoning. In the latter case, we have devised a hybrid algorithm where the results of the FSR are used to refine the initial summary to isolate known solutions, and the rest of refinement proceeds normally to find any remaining solutions to the query. In this section, we discuss the two FSRs we have implemented and provide an overview of the hybrid algorithm.

3.1 Query Expansion

Query expansion is a well-known technique to find implicit solutions to a query by expanding the query (in effect, producing a union of conjunctive queries) based on axioms in the ontology, e.g., expanding a membership query $C(x)$ by adding query atoms for all subclasses of C in the ontology. QuOnto [7] is one such implementation of a query expansion algorithm which is sound and complete for the logic DL-Lite. Our query expansion algorithm is similar in spirit to QuOnto, however, we differentiate ourselves in a few ways. First, we use an OWL-DL reasoner to compute explicit plus inferred subclasses for a concept in the query and use these as a basis for expansion. The inferred results found by the reasoner helps the algorithm cover more cases. Second, we use a Datalog engine to compute *same* individuals in the ABox based on deterministic mergers (i.e. inferences due to functional and inverse-functional property axioms, and sameAs axioms), and use the results to expand our solution set. Finally, since query expansion can produce a large number of queries, we eliminate queries that have no solutions by checking for potential matches in the summary ABox (e.g, if the query contains atom $C(x)$ but the concept assertion $C(a)$ is not present in the summary ABox, we discard the query as it cannot have a match in the original ABox). Details of our query expansion algorithm are provided in a related report [8]. Note that our algorithm is sound and complete for $\mathcal{DL}$-Lite, and for the logic $\mathcal{EL}$ when the KB is acyclic, and is otherwise incomplete.

3.2 $\mathcal{EL}+$ Reasoner

We have implemented a polynomial-time sound and complete $\mathcal{EL}+$ reasoner based on the algorithm described in [9]. Note that many ontologies in the health-care and life-sciences community such as the Gene Ontology (GO) fall in this fragment, which can express (among other things) general concept inclusions, sub-roles and transitive relations.

We have two separate implementations – an highly optimized in-memory version, and a database-backed version that uses a Datalog engine to evaluate the $\mathcal{EL}+$ rules. In our experiments, the in-memory $\mathcal{EL}+$ reasoner has been able to fully classify the SNOMED OWL ontology in under 15 mins. Also, the in-memory version supports incremental updates and provides explanations for subsumption (i.e., smallest set of axioms responsible for the subsumption).

3.3 Hybrid Reasoner

The main idea of our hybrid reasoning approach is that it can incorporate *any* sound and incomplete reasoning algorithm (or FSR) into the core SHER summarization and refinement process to provide efficient, complete, and yet highly scalable reasoning over large Aboxes. The key insight is that the solutions from the FSR can be used as a partitioning function for refinement instead of partitioning based on edges. This effectively removes the obvious solutions from the summary Abox. If the FSR finds all solutions, there will be no solutions left in the summary Abox after this first refinement, so the algorithm will converge very quickly. Any remaining inconsistencies are spurious, and can be resolved in one or a few refinement steps. If the FSR finds only some of the solutions, then the refinement process will find the rest of the solutions with fewer refinement steps. We have demonstrated the value of this hybrid approach in our Clinical Trials use case (described in the next section), where we achieved noticeable improvement in performance using our query expansion algorithm as the FSR, and summarization/refinement to find only remaining solutions.

4 Use Cases

We used SHER to build two solutions within the healthcare and life sciences domain: the first used SHER to semantically query patient records using SNOMED-CT, and the second used SHER to provide a semantic search capability over the medical literature from the National Library of Medicine. Each of these use cases is described below.

4.1 Semantic Querying of Patient Records

In collaboration with Columbia University Medical Center, we used SHER to design a solution for the problem of finding matching patient records for clinical trials criteria [10]. Currently, there are approximately 65,000 clinical trials that are designed to test various drugs and procedures. Each clinical trial posts the criteria for recruitment on a central website `http://clinicaltrials.gov`. Common criteria for recruitment include patients being on drugs with certain active ingredients, or patients being diagnosed with various medical conditions. Electronic patient records contain vendor-specific drugs that a patient is taking, or specific radiological or laboratory findings; they never contain the ingredients of the drugs that patients are on, nor do they contain detailed diagnostic medical conditions[2]. In order to find patient records that satisfy the clinical trials criteria, we need to bridge the semantic gap between, for example, the vendor-specific drug information that is part of the patient record, and the specific active ingredients of the drug that is not part of the record, but that is part of the domain-specific knowledge of medicine. We investigated if we could bridge

[2] Medical diagnoses recorded for billing purposes do not necessarily satisfy the granularity needed for clinical trials recruitment

this semantic gap using SNOMED-CT as our knowledge base, and SHER as the reasoning engine. There were three key technical challenges in building this solution:

- (a) Transforming the patient data into a SNOMED-CT knowledge base. This step included mapping the local terminology of Columbia called MED into SNOMED-CT.
- (b) Scaling reasoning to SNOMED-CT, with a very large Abox (1 year worth of patient data for 250K patients at Columbia, which was about 60 million RDF triples).
- (c) The expressivity of the ontology, which was $\mathcal{SHIN}$ due to negation in the Abox (e.g., we had to model complex negation where a certain laboratory finding was ruled out).

In our initial results, the performance of our initial solution on nine queries drawn from `http://clinicaltrials.gov` ranged between 26-370 minutes, because of the expressivity of the knowledge base. Although this performance is acceptable within a domain where the clinical trial matching is largely a manual and tedious process, it is less than ideal for other use case scenarios. We gained a significant performance improvement by adding a Fast Sound Reasoner (FSR) component into our system to find as many solutions as quickly as possible. With the addition of the FSR, our performance improved to between 11-21 minutes for the same data (see [8] for details). For the 9 test queries, the FSR component found all answers. Although our knowledge base was expressive, and we were in general not complete for all queries, for the specific sets of queries we used, the FSR component was sufficient in providing fast answers to the question. In fact, because FSR uses nothing more than a SQL expansion of the query, performance for just the FSR component is significantly under the 11-21 minutes range reported in [8]; the majority of the 11-21 minutes is spent in summarization and refinement steps post FSR to check for completeness.

Some lessons learnt from this use case include (a) the expressivity of the knowledge base is not in the simpler subsets of OWL-DL, but user expectation is nevertheless that the system perform like a database (i.e., queries must return in milliseconds), (b) there is a need for both closed world and open world reasoning on the same dataset, an issue we did not really address, (c) there is a need to combine rules and ontologies, but most would fall into DL-safe rules.

4.2 Semantic Search over the Medical Literature: AnatomyLens

In searching the biomedical literature, using the concepts in an ontology can help improve recall. For instance, if users want to search for medical articles that mention the heart, they implicitly mean articles that either mention the heart or any of its subparts. An ontology like the Foundational Model of Anatomy (FMA) formally defines part hierarchies that can be used to automatically improve recall of medical data. Similarly, when users want to find genes that are involved in a certain biological process such as neuron development, they implicitly mean any

subprocesses of neuron development (such as axon or dendrite development). Ontologies (such as the Gene Ontology (GO)), which formally describe these subprocesses, can be used to improve recall of relevant genes or articles. Anatomy Lens[3] is a SHER-based solution for semantic search of the medical literature using ontologies such as GO, FMA, and MeSH.

The data set used in this solution was based in part on the Banff HCLS Demo[4]. We integrated the following data into one large knowledge base with 300 million RDF triples:

- PubMed 2008 distribution from NLM (National Library of Medicine) with only article titles and their links to MeSH.
- Gene annotations GOA, linking genes and gene products to articles, specific evidence codes, and gene ontology processes (such as dendrite development) defined in the Gene Ontology.
- The Gene Ontology, which contains definitions of the biological, cellular, and molecular functions of genes
- The Foundational Model of Anatomy, which contains definitions of anatomical parts and their subparts. We used the OWL version of FMA created by Golbreich, Zhang, Bodenreider (2006).
- Mappings from the MeSH annotations for PubMed articles to FMA concepts. This was achieved by using UMLS to map MeSH to FMA, but it missed some key mappings. We augmented the mappings with additional matches from the MMTx tool from NLM `http://mmtx.nlm.nih.gov/index.shtml`, keeping only matches with a perfect score.
- The MeSH taxonomy, which is really three separate trees. For example, the concept *Program Evaluation* appears in three trees, but it contains the subclass *Benchmarking* in only two of the three trees. To be consistent in our reasoning, we treated these separate trees as a directed acyclic graph.

The four key technical challenges in this solution were as follows:

- It is well known that certain combinations of constructors are particularly problematic for OWL reasoning. An example of this may be seen in reasoning on FMA. FMA is a deep partonomy, with both *hasPart* and *partOf* relations and an inverse relation between *partOf* and *hasPart*. This particular combination of constructs causes reasoners to fail (see `http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/index.html`). We therefore included only *partOf* relations in FMA, as is recommended. This version of FMA falls into the OWL dialect $\mathcal{EL}+$, and we used the $\mathcal{EL}+$ reasoner in SHER to reason on this dataset.
- Even with the $\mathcal{EL}+$ reasoner, scalability was a serious challenge for a Web service. The combined Tbox for AnatomyLens has 75,000 FMA concepts, 30,000 GO concepts, and 15,000 MeSH concepts. Reasoning at query time on this combined Tbox took a few minutes with the $\mathcal{EL}+$ reasoner, which is

[3] http://services.alphaworks.ibm.com/anatomylens/
[4] http://esw.w3.org/topic/HCLS/Banff2007Demo

not acceptable for a web based application. We exploited the fact that the Abox in AnatomyLens does not contain any relations which could be used to infer new types. We therefore precomputed all subclasses and subparts for each concept in advance, and used this expansion to expand the query for each concept. For any given queried concept C, we translate the query into a membership query where we look for all instances of the concept $(\exists partOf.C) \sqcup C$.

- A key issue in semantic search is to be able to rank the returned results. In keyword search, it is obvious what constitutes a match, and there are various techniques to rank the result set. For semantic search, one measure that can be used to rank the results is the semantic distance between an answer concept Q and the queried concept C. Our metric of semantic distance was the size of the justification set for any given concept Q which was an answer to the query $(\exists partOf.C) \sqcup C$ (i.e., the more the number of axioms needed to infer a relationship between Q and the query, the greater the semantic distance between them). We used this metric to rank the returned results, and grouped each set of results by justifications. We also provided the natural language descriptions of the justification sets to enable users to understand why a set of results were a match. The ability to provide explanations for matches is critical in semantic search, where the link between the query and the returned results may not always be obvious to the user.
- Another issue in query-expansion based semantic search is dealing with peculiar modeling issues. For example, in the partonomy in FMA, the concept *Cell* is a transitive sub-part of each body part, but including articles that talk about cells when the query is about a specific body part such as the *Lung* would produce a lot of extraneous results. To solve this problem, we ensure that the expanded sub-concept is *dominated* by the queried concept in the partonomy tree. Finally, we allow the user to control the expansion by presenting results closest in semantic distance to the queried concept, and prompting users for whether they want to continue to expand the semantic search.

Lessons learnt from this use case include (a) difficulty in modeling constructs with existentials, when in fact, what is needed is more analogous to a description graph [11], (b) difficulty in applying generic ontologies to semantic search, (c) need for a combination of OWL as well as IR techniques to improve coverage in semantic search, (d) need for expressive reasoning over Aboxes, but inexpressive reasoning on Tboxes.

4.3 Information retrieval over unstructured text

A key problem in statistical natural language processing is that it is frequently error prone. It is not uncommon for machine learners for example to make errors in typing *George Bush International Airport* as a person. Yet, such typing is critical to organizing the vast web of unstructured information into structured, computable information. We explored the use of OWL ontologies in quickly

discovering such typing errors in statistical natural language processing [12]. The core idea was to exploit simple domain and range constraints along with disjointness axioms to locate typing errors. As an example, if the *George Bush International Airport* is associated with a property such as *is hub of*, then from a domain restriction, one can infer that it is an airport. Furthermore, because *Airport* and *Person* are disjoint, this would be isolated as a typing error. A problem in applying such techniques in a large scale to text is the difficulty in creating and maintaining such ontologies. We've explored using simple statistics on large datasets such as Freebase and DBpedia to automatically construct large sets of fuzzy domain, range, and disjointness axioms, and explored the validity of applying these constraints to the linked open dataset to infer new types and pinpoint inaccuracies in typing in Freebase or DBpedia [13]. Most of this type of reasoning is however relatively inexpressive, and probabilistic in nature.

4.4 Master Data Management

Product data tends to be organized in deep hierarchies, with many different hierarchies to represent derived attributes of the data. As one example, retailers organize products into location hierarchies (often to the point of a particular store on a shelf), and often want to inherit attributes for products, but also be able to override these attributes (e.g., override the default price for a particular location). OWL reasoning did not really fit the requirements of master data management.

5 Conclusions

At least in the use cases we have seen, the emphasis in real world appears to be more in terms of fast, highly scalable reasoning over relatively inexpressive knowledge bases. When more expressive reasoning is required, it appears that the pragmatic expectations of database-like queries outweigh need for completeness. Furthermore, many of the more expressive ontologies seem to use existentials to model elements that might be better modeled using description graphs.

References

1. Fokoue, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: The summary abox: Cutting ontologies down to size. In: International Semantic Web Conference (ISWC). (2006) 343–356
2. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: AAAI. (2007) 299–304
3. Lu, J., Ma, L., Zhang, L., Brunner, J.S., Wang, C., Pan, Y., Yu, Y.: Sor: A practical system for ontology storage, reasoning and search. In: VLDB. (2007) 1402–1405
4. Horrocks, I., Patel-Schneider, P.F.: Reducing owl entailment to description logic satisfiability. In: Journal of Web Semantics, Springer (2003) 17–29

5. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. J. Web Sem. **5**(2) (2007) 51–53
6. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of owl dl entailments. In: ISWC/ASWC. (2007) 267–280
7. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. (2005) 1670–1671
8. Dolby, J., Fokoue, A., Kalyanpur, A., Ma, L., Patel, C., Schonberg, E., Srinivas, K., Sun, X.: Efficient reasoning on large shin aboxes in relational databases. Technical report, IBM Research (2008)
9. Baader, F., Lutz, C., Suntisrivaraporn, B.: Is tractable reasoning in extensions of the description logic el useful in practice? In: Proceedings of the 2005 International Workshop on Methods for Modalities (M4M-05). (2005)
10. Patel, C., Cimino, J.J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Matching patient records to clinical trials using ontologies. In: ISWC/ASWC. (2007) 816–829
11. Motik, B., Grau, B.C., Horrocks, I., Sattler, U.: Representing structured objects using description graphs. In: KR. (2008) 296–306
12. Dolby, J., Fan, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Murdock, J.W., Srinivas, K., Welty, C.A.: Scalable cleanup of information extraction data using ontologies. In: ISWC/ASWC. (2007) 100–113
13. Dolby, J., Fokoue, A., Kalyanpur, A., Srinivas, K., Schonberg, E.: Extracting enterprise vocabulary using linked open data. `http://domino.watson.ibm.com/library/Cyberdig.nsf/papers/4D84639C32795569852574FD005EA539/` (2008)