Arachne: an OWL RL reasoner applied to Gene Ontology Causal Activity Models (and beyond)

James P. Balhoff[1], Benjamin M. Good[2], Seth Carbon[2], Christopher J. Mungall[2]

[1] Renaissance Computing Institute, University of North Carolina, Chapel Hill, NC 27517, USA
[2] Lawrence Berkeley National Lab, Berkeley, CA 94720, USA
balhoff@renci.org, {bgood, sjcarbon, cjmungall}@lbl.gov

Abstract. This paper introduces Arachne, an RDF rule engine with support for efficient reasoning with large OWL RL terminologies. Arachne is being used by the Gene Ontology (GO) Consortium to provide real-time reasoning within the Noctua modeling tool while creating GO "Causal Activity Models".

1 Introduction

Of the hundreds of ontologies in use in biology, the Gene Ontology (GO) [1] is perhaps the best known and most widely used. The GO is used to classify genes found in the genome of any given species based on the function of the 'molecular machine' encoded by that gene. It captures knowledge about the function of gene products in terms of their localizations within the cell, the molecular functions they enable, and the biological processes that they help to carry out. Historically, this ontology has been used to 'tag' gene products with labels from the classes in the ontology. Reasoning has primarily been applied to validate the consistency of the class hierarchy and to infer additional subsumption relationships [2].

Now, the GO consortium is shifting to a new knowledge representation paradigm dubbed 'causal activity models' (GO-CAM). As opposed to simply associating gene products with classes from the ontology, each GO-CAM provides a detailed semantic model of how one or several gene products contribute to the execution of a biological process. GO-CAMs are implemented with the OWL 2 Web Ontology Language [3]. OWL individuals are used to represent the nodes in the model (corresponding to genes, functions, etc.). Each individual is typed with a class or classes from the Gene Ontology and related ontologies such as ChEBI [4], and linked to other individuals via properties selected from the OBO Relations Ontology [5]. These models are constructed by professional knowledge engineers ('curators') in a web stack named Noctua (https://github.com/geneontology/noctua). Noctua provides a rich multi-user graphical client on top of a knowledge system backed with an RDF triplestore.

Each GO-CAM is a set of assertions about OWL individuals (an 'Abox'). For example, Fig. 1 shows a representation of knowledge about a protein complex, located in the nucleus, that is involved in enabling DNA polymerase activity. The OWL definitions for the classes and relationships used to make these assertions comprise the 'Tbox'.

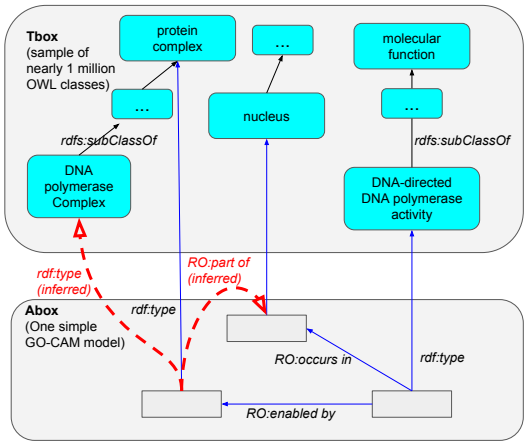

Fig. 1. Capturing knowledge with the Gene Ontology.

All GO-CAMs are modeled using the same Tbox, which contains ~2 million logical axioms and nearly one million classes. By applying axioms from the Tbox to the instance graph in the Abox, additional statements can be inferred—for example, for the Abox in Fig. 1, that the instance of 'protein complex' is also an instance of the more specific 'DNA polymerase complex'. Upon this semantic backdrop, the Noctua application needs to:

1. Ensure that the models generated are logically consistent.
2. Provide access to statements that are not explicitly declared in the model but can be inferred.
3. Provide explanations for inferred statements.
4. Allow for collaborative, simultaneous model editing: reasoning needs to perform quickly enough to be integrated into the editing experience.

The Gene Ontology project uses the ELK reasoner for the OWL EL profile [6] during ontology development, and while it has been transformative for tasks such as ontology classification and consistency checking, it was not a good fit for a real-time multi-user online system focused on graphs of instance data. It does not support some types of axioms needed in the context of Abox graph reasoning: inverse properties, property ranges, and materialization of object property assertions. Also, while ELK supports incremental classification, it answers a single query at a time. It does not support preclassification of Tbox inferences and then concurrent extension to process multiple independent datasets.

2 Arachne RDF rule engine for OWL RL

To support our need to simultaneously reason over any number of data models, as well as quickly materialize all inferred instance relationships for each, we developed the

Arachne RDF rule engine for the RL subset of OWL (https://github.com/balhoff/arachne). OWL RL provides expressive reasoning on property relationships, and can be implemented using rule-based technologies [7]. Arachne is a forward-chaining rule engine, implemented in Scala and based on the Rete/UL algorithm as described by Doorenbos [8]. This algorithm allows efficient matching of relevant rules even when the rule set is extremely large. A separate component of Arachne is an OWL API-based translator which converts OWL Tbox axioms to corresponding rules from the OWL RL profile (https://github.com/balhoff/owl-to-rules). Arachne currently supports all OWL RL constructs, with the exception of HasKey and reasoning with data properties. A subset of SWRL [9] is also supported. Rather than using a fixed OWL RL ruleset operating on the Tbox as RDF triples, the translator converts Tbox axioms directly into rules that can be used to efficiently derive only Abox conclusions (i.e., inferred class assertions and object property assertions), similar to the approach used by RDFox [10]. For example, given the OWL axiom SubClassOf('nucleus', 'organelle'), a rule such as the following is generated:

(?x rdf:type 'nucleus') → (?x rdf:type 'organelle')

Arachne translates the Tbox ontology used in Noctua into more than 1 million rules, only a limited set of which may apply to any given instance model. This preprocessing, which takes ~60 seconds on a 2017 Apple MacBook Pro, can be performed at application startup, producing an immutable rule engine which can be used to concurrently materialize inferences for any number of models. For a typical model consisting of hundreds of triples, inference materialization typically takes around 1–2 seconds, producing 1000–2000 additional triples.

Before implementing Arachne, we explored integration of two other RDF rule engines into Noctua. The first, RDFox, is an extremely fast reasoner that scales well to large numbers of rules and input triples [10]. However, it is meant to work as a single triplestore, rather than being used to reason over many independent datasets. RDFox is implemented in C++ with a Java API wrapper, making cross-platform packaging and distribution more complicated than desired. Finally, we determined that the academic license which applies to RDFox was too restrictive for integration and distribution with our free and open-source stack.

The second rule engine we explored was the forward-chaining reasoner included with the Apache Jena RDF API [11]. We found that an initial integration of Jena into Noctua worked well: with small to intermediate numbers of rules (e.g., 1362 rules derived from the OBO Relations Ontology) it is as fast or faster than Arachne. However, as the number of rules increased, performance degraded significantly. For example, using 1,006,200 rules derived from the full GO-CAM Tbox, after the initial loading of rules the Jena implementation requires ~250 seconds to derive a total of 2257 triples from a starting point of 412 triples, while Arachne requires ~1 second.

The addition of the Arachne reasoner to Noctua allows curators to work while the reasoner constantly runs in the background, dynamically making new inferences and checking for consistency. Nodes on the canvas are labeled with any inferred 'direct' types in addition to their asserted types, and warnings are raised when errors are intro-

duced. Curators can also explore an Inference Explanations view, which details the asserted triples and reasoning rules leading to each inferred statement. The fully materialized set of inferences is used within Noctua to support SPARQL-driven tabular data exports. As we continue to develop the Noctua interface, we intend to more deeply integrate reasoning results into the editing experience to serve as a kind of logical spell checker—always on, offering corrections as statements are created.

3 Conclusion

Arachne provides a convenient implementation of OWL RL reasoning which can be readily integrated into Java-based software or used via its standalone command-line interface. Its architecture is well suited to applications of large ontologies to instance-based datasets, supporting new approaches to utilizing the rich semantics represented within the Gene Ontology and related scientific ontologies. Arachne is open source and available under a BSD license.

Acknowledgments. We would like to thank N. Harris for help with editing, and D. Osumi-Sutherland, K. Van Auken, D. Hill, P. Thomas, and other GO Consortium members for feedback on reasoning results. Arachne and Noctua development is supported by NIH-NHGRI 2U41HG002273-17.

References

1. The Gene Ontology Consortium: Expansion of the Gene Ontology knowledgebase and resources. Nucleic Acids Res. 45, D331–D338 (2017).
2. Mungall, C.J., Dietze, H., Osumi-Sutherland, D.: Use of OWL within the Gene Ontology. Proceedings of OWLED 2014, Riva del Garda, Italy (2014).
3. OWL 2 web ontology language: Structural specification and functional-style syntax, https://www.w3.org/TR/owl2-syntax/.
4. Hastings, J., de Matos, P., Dekker, A., Ennis, M., Harsha, B., Kale, N., Muthukrishnan, V., Owen, G., Turner, S., Williams, M., Steinbeck, C.: The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. Nucleic Acids Res. 41, D456–63 (2013).
5. Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A.L., Rosse, C.: Relations in biomedical ontologies. Genome Biol. 6, R46 (2005).
6. Kazakov, Y., Krötzsch, M., Simančík, F.: The Incredible ELK. J. Automat. Reason. 53, 1–61 (2013).
7. OWL 2 Web Ontology Language Profiles (Second Edition), https://www.w3.org/TR/owl2-profiles/.
8. Doorenbos, R.B.: Production Matching for Large Learning Systems, http://reports-archive.adm.cs.cmu.edu/anon/1995/CMU-CS-95-113.pdf, (1995).
9. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, https://www.w3.org/Submission/SWRL/.
10. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: RDFox: A Highly-Scalable RDF Store. In: The Semantic Web - ISWC 2015. pp. 3–20. Springer International Publishing (2015).
11. Apache Jena, https://jena.apache.org.