Navigating OWL 2 Ontologies through Graph Projection*

Ahmet Soylu[1,2] and Evgeny Kharlamov[3]

[1] Norwegian University of Science and Technology, Gjøvik, Norway
ahmet.soylu@ntnu.no
[2] SINTEF Digital, Oslo, Norway
[3] University of Oxford, Oxford, the UK
evgeny.kharlamov@cs.ox.ac.uk

Abstract. Ontologies are powerful, yet often complex, assets for representing, exchanging, and reasoning over data. Particularly, OWL 2 ontologies have been key for constructing semantic knowledge graphs. Ability to navigate ontologies is essential for supporting various knowledge engineering tasks such as querying and domain exploration. To this end, in this short paper, we describe an approach for projecting the non-hierarchical topology of an OWL 2 ontology into a graph. The approach has been implemented in two tools, one for visual query formulation and one for faceted search, and evaluated under different use cases.

Keywords: OWL 2 · Ontologies · Graph navigation · Knowledge graphs.

1 Introduction

Ontologies are powerful, yet often complex, assets for representing, exchanging, and reasoning over data. Particularly, OWL 2 ontologies [4] have been key for constructing semantic knowledge graphs (e.g., [7,8]). A knowledge graph describes real world entities and their interrelations [17]. They have been used both in academia, such as Yago [3] and DBpedia [9], and in industry such as Google's Knowledge Graph, Facebook's Graph Search, and Microsoft's Satori. Semantic knowledge graphs are typically stored or exported as RDF datasets, which allow for storing sparse and diverse data in an extensible and adaptable way [16]. Semantics of such datasets are typically encoded in OWL 2 ontologies.

Ability to navigate ontologies is essential for understanding the domain of interest (e.g., visual exploration) [6,10], its representation, and underlying data; and for supporting various other knowledge engineering tasks such as querying (e.g., query by navigation) [12,14]. However, it is not straight forward to explore implicit and explicit connections between the classes of an OWL 2 ontology, which is basically a collection of logical axioms. To this end, in this short paper, we describe an approach for projecting the non-hierarchical class topology of an OWL 2 ontology into a graph. This approach has been implemented in two semantic

* Funded by EU H2020 TheyBuyForYou (780247) and FP7 Optique (318338) projects.

tools, namely OptiqueVQS[11] for visual query formulation and SemFacet [1] for faceted search and evaluated under different use cases.

The rest of the paper is structured as follows. Sect. 2 presents our graph projection approach from ontologies, while Sect. 3 presents the tools using our approach. Finally, Sect. 4 concludes the paper.

2 Graph Projection

Our goal for graph projection is, given an ontology, to create a directed labelled graph, called navigation graph [1], whose nodes correspond to the named classes and datatypes in the ontology and edges between nodes to the object properties and datatype properties. Let C_1, C_2, and C_3 be classes, r_1, r_2, and r_3 object properties, d_1 a datatype property, i_1 and i_2 individuals, and dt_1 a data type. First, each class and datatype in the ontology is translated to a node in the navigation graph. Then we add edges of the form $r_1(C_1, C_2)$ and $d_1(C_1, dt_1)$ derived from the axioms of the ontology. The types of axioms resulting in an edge are presented with examples in the followings using description logic (DL) [2].

Ontologies have a propagative effect on the amount of information to be presented. This case is considered in two forms, namely the top-down and bottom-up propagation of property restrictions [5,14]. The first form emerges from the fact that, in an ontology, explicit restrictions attached to a class are inherited by its subclasses. The second form is rooted from the fact that the interpretation of an OWL class also includes the interpretations of all its subclasses. Therefore, for a given class, it may also make sense to derive edges from the (potential) object and datatype properties of its subclasses and superclasses.

2.1 Edges through Object Properties

Domains and Ranges: Domain and range axioms using named classes are translated to an edge. For example, axioms given in Ex. 1 map to edge $r_1(C_1, C_2)$.

$$\exists r_1.\top \sqsubseteq C_1 \text{ and } \top \sqsubseteq \forall r_1.C_2 \tag{1}$$

$$\exists r_1.\top \sqsubseteq C_1 \text{ and } \top \sqsubseteq \forall r_1.(C_2 \sqcup C_3) \tag{2}$$

If a complex class expression, formed through intersection ($\sqcap$) or union ($\sqcup$), appears as a domain and/or range, then an edge is created for each pair of domain and range classes. For example, axioms given in Ex. 2 map to edges $r_1(C_1, C_2)$ and $r_1(C_1, C_3)$.

Object Property Restrictions: Object property restrictions used in class descriptions, formed through existential quantification ($\exists$), universal quantification ($\forall$), individual value restriction, max ($\geq$), min ($\leq$), and exactly ($=$), are mapped to edges. For example, axioms given in Ex. 3 to 5 map to $r_1(C_1, C_2)$. Note that in Ex. 5, there is a complex class expression on the left-hand-side.

$$C_1 \sqsubseteq \exists r_1.C_2 \tag{3}$$

$$C_1 \equiv \leq_n r_1.C_2 \tag{4}$$

$$\forall r_1.C_1 \sqsubseteq C_2 \tag{5}$$

Axioms given in Ex. 6 include an individual value restriction and an edge is created with the type of individual, that is $r_1(C_1, C_2)$.

$$C_1 \sqsubseteq \exists r_1.\{i_1\} \text{ , and } i_1 : C_2 \tag{6}$$

Axiom given in Ex. 7 includes a complex class expression. In this case, an edge is created for each named class, that is $r_1(C_1, C_2)$ and $r_1(C_1, C_3)$.

$$C_1 \sqsubseteq \exists r_1.(C_2 \sqcup C_3) \tag{7}$$

Given an enumeration of individuals, an edge is created for each individual's type. For example, axioms given in Ex. 8 map to two edges, that is $r_1(C_1, C_2)$ and $r_1(C_1, C_3)$.

$$C_1 \sqsubseteq \exists r_1.\{i_1\} \sqcup \{i_2\} \text{ , } i_1 : C_2 \text{ , and } i_2 : C_3 \tag{8}$$

Inverse Properties: Given an edge in the navigation graph such as $r_1(C_1, C_2)$ and an inverse property axiom for the corresponding object property such as given in Ex. 9, a new edge is created for the inverse property, that is $r_1^-(C_2, C_1)$.

$$r_1 \equiv r_1^- \tag{9}$$

Role Chains: Given two edges $r_1(C_1, C_2)$ and $r_2(C_2, C_3)$ in the navigation graph, and a role chain axiom between r_1, r_2, r_3 such as given in Ex. 10, a new edge is created for $r3$, that is $r_3(C_1, C_3)$.

$$r_1 \circ r_2 \sqsubseteq r_3 \tag{10}$$

Top-down Propagation: Given an edge $r_1(C_1, C_2)$ in the navigation graph and a subclass axiom such as as given in Ex. 11, a new edge is added to the graph, that is $r_1(C_3, C_2)$. Similar edges could be created for subproperties.

$$C_3 \sqsubseteq C_1 \tag{11}$$

Bottom-up Propagation: Given an edge $r_1(C_1, C_2)$ in the navigation graph and a subclass class axiom such as given in Ex. 12, a new edge is added to the graph, that is $r_1(C_3, C_2)$. Similar edges could be created for superproperties.

$$C_1 \sqsubseteq C_3 \tag{12}$$

2.2 Edges through Datatype Properties

Domains and Ranges: Domain and range axioms using datatype properties are translated to an edge. For example, axioms given in Ex. 13 map to an edge, that is $d_1(C_1, dt_1)$.

$$\exists d_1.DatatypeLiteral \sqsubseteq C_1 \text{ and } \top \sqsubseteq \forall r_1.dt_1 \tag{13}$$

Datatype Property Restrictions: Datatype property restrictions, formed through existential quantification ($\exists$), universal quantification ($\forall$), max ($\geq$), min ($\leq$), exactly ($=$), and value are mapped to edges. For example, axiom given in Ex. 14 maps to $d_1(C_1, dt_1)$.

$$C_1 \sqsubseteq \exists d_1.dt_1 \tag{14}$$

Top-down Propagation: Given an edge $d_1(C_1, dt_1)$ in the navigation graph and a subclass axiom such as as given in Ex. 15, a new edge is added to the graph, that is $d_1(C_2, dt_1)$. Similar edges could be created for subproperties.

$$C_2 \sqsubseteq C_1 \tag{15}$$

Bottom-up Propagation: Given an edge $d_1(C_1, dt_1)$ in the navigation graph and a subclass class axiom such as given in Ex. 16, a new edge is added to the graph, that is $d_1(C_3, dt_1)$. Similar edges could be created for superproperties.

$$C_1 \sqsubseteq C_3 \tag{16}$$

3 Applications

Variants of this approach have been implemented and evaluated in OptiqueVQS [11], a visual query formulation tool, and SemFacet [1], a faceted search tool. Both interfaces support tree-shaped conjunctive queries.

OptiqueVQS (see Fig. 1) is a visual query system. It allows users to navigate the conceptual space and each traversal from a class to another adds a typed variable-node and object property connecting it to the query graph. OptiqueVQS was deployed and evaluated in different use cases, including Siemens' case for sensor data [15], Statoil's case for oil and gas [11], and on generic datasets [13]. SemFacet (see Fig. 2) is full-fledged general-purpose faceted search interface. In typical faceted search, users are presented with facet-values organised in groups according to facet-names and it is often not allowed to navigate between classes. SemFacet allows end users to navigate between classes and browse data sets at the same time. The interface was deployed and evaluated over a slice of Yago database [1].

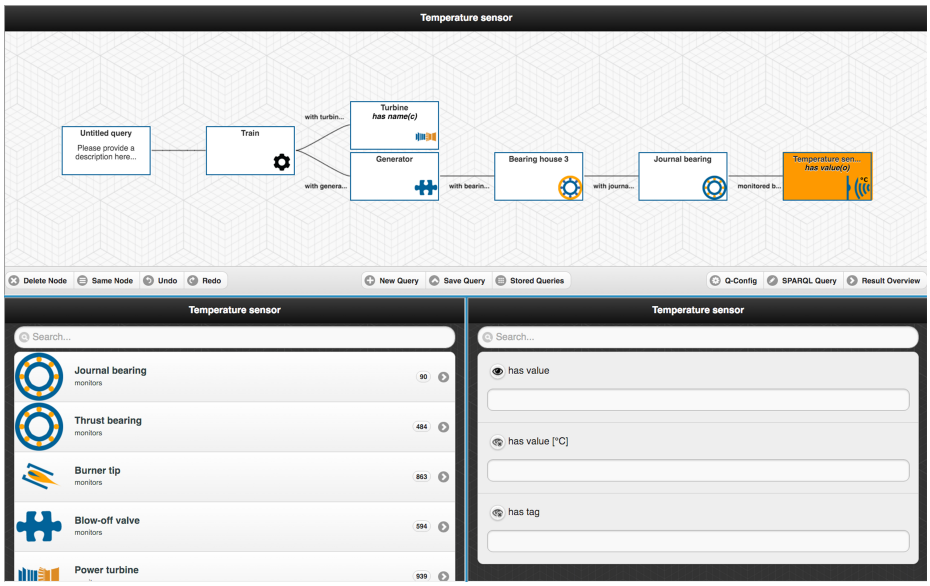

Fig. 1. OptiqueVQS over a use case provided by Siemens.

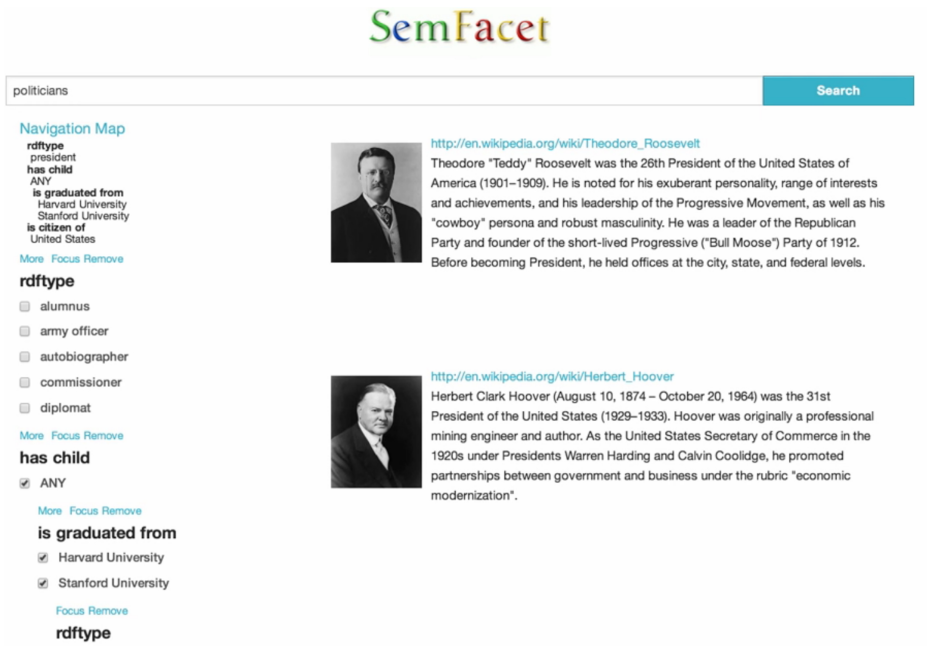

Fig. 2. SemFacet over Yago database.

4 Conclusions

In this paper, we presented an approach, together with two example applications, for navigating OWL 2 ontologies by projecting them into graphs through harvesting a set of axioms. A future challenge is to enable users to navigate distant classes that are not directly connected but are multiple edges away. We call this non-local navigation, which could be useful for navigating large class networks.

References

1. Arenas, M., et al.: Faceted search over RDF-based knowledge graphs. Journal of Web Semantics **37-38**, 55–74 (2016)
2. Baader, F., et al. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA (2003)
3. Biega, J., et al.: Inside YAGO2s: A Transparent Information Extraction Architecture. In: Proceedings of the 22nd International Conference on World Wide Web (WWW 2013). pp. 325–328. ACM, New York, NY, USA (2013)
4. Grau, B.C., et al.: OWL 2: The next step for OWL. Journal of Web Semantics **6**(4), 309–322 (2008)
5. Grau, B.C., et al.: Towards Query Formulation, Query-Driven Ontology Extensions in OBDA Systems. In: Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) (2013)
6. Katifori, A., et al.: Ontology Visualization Methods – a Survey. ACM Computing Surveys **39**(4) (2007)
7. Kharlamov, E., et al.: Ontology Based Data Access in Statoil. Journal of Web Semantics **44**, 3–36 (2017)
8. Kharlamov, E., et al.: Semantic access to streaming and static data at Siemens. Journal of Web Semantics **44**, 54–74 (2017)
9. Lehmann, J., et al.: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web **6**(2), 167–195 (2015)
10. Lohmann, S., et al.: Visualizing ontologies with VOWL. Semantic Web **7**(4), 399–419 (2016)
11. Soylu, A., et al.: OptiqueVQS: a Visual Query System over Ontologies for Industry. Semantic Web **(to appear)**
12. Soylu, A., et al.: Ubiquitous web navigation through harvesting embedded semantic data: A mobile scenario. Integrated Computer-Aided Engineering **19**(1), 93–109 (2012)
13. Soylu, A., et al.: Experiencing OptiqueVQS: a multi-paradigm and ontology-based visual query system for end users. Universal Access in the Information Society **15**(1), 129–152 (2016)
14. Soylu, A., et al.: Ontology-based end-user visual query formulation: Why, what, who, how, and which? Universal Access in the Information Society **16**(2), 435–467 (2017)
15. Soylu, A., et al.: Querying industrial stream-temporal data: An ontology-based visual approach. Journal of Ambient Intelligence and Smart Environments **9**(1), 77–95 (2017)
16. Suchanek, F.M., et al.: Knowledge Bases in the Age of Big Data Analytics. Proceedings of the VLDB Endowment **7**(13), 1713–1714 (2014)
17. Yan, J., et al.: A Retrospective of Knowledge Graphs. Frontiers of Computer Science **12**(1), 55–74 (2018)