# Uses and Control of Inferencing in Knowledge Graphs

**by Mike Bergman - Wednesday, March 15, 2017**

http://www.mkbergman.com/2030/uses-and-control-of-inferencing-in-knowledge-graphs/


## Dialing In Queries from the General to the Specific

Inferencing is a common term heard in association with semantic technologies, but one that is rarely defined and still less frequently described as to value and rationale. I try to redress this gap in part with this article.

Inferencing is the drawing of new facts, probabilities or conclusions based on reasoning over existing evidence. Charles Sanders Peirce classed inferencing into three modes: deductive reasoning, inductive reasoning and abductive reasoning. Deductive reasoning extends from premises known to be true and clear to infer new facts. Inductive reasoning looks at the preponderance of evidence to infer what is probably true. And abductive reasoning poses possible explanations or hypotheses based on available evidence, often winnowing through the possibilities based on the total weight of evidence at hand or what is the simplest explanation. Though all three reasoning modes may be applied to knowledge graphs, the standard and most used form is deductive reasoning.

An inference engine may be applied to a knowledge graph and its knowledge bases in order to deduce new knowledge. Inference engines apply either backward- or forward-chaining deductive reasoning. In backward chaining, the reasoning tests are conducted "backwards" from a current consequent or "fact" to determine what antecedents can support that conclusion, based on the rules used to construct the graph. ("What reasons bring us to this fact?") In forward chaining the opposite occurs; namely, a goal or series of goals are stated and then existing facts (as rules) are checked to see which ones can lead to the goal. (" A goal X may be possible because of?") The process is iterated until the goal is reached or not; if reached, new knowledge in terms of heretofore unstated connections may be added to the knowledge base.

Inference engines can be applied at the time of graph building or extension to test the consistency and logic of the new additions. Or, semantic reasoners may be applied to a current graph in order to expand queries for semantic search or for these other reasoning purposes. In the case of Cognonto's KBpedia knowledge structure, which is written in OWL 2, though the terminology is slightly different, the

groundings are in first-order logic (FOL) and description logics. These logical foundations provide the standard rules by which reasoners can be applied to the knowledge graph [1]. In this article, we will not be looking at how inferencing is applied during graph construction, a deserving topic in its own right. Rather, we will be looking at how inferencing may be applied to the existing graph.

## Use of Reasoning at Run Time

Once a completed graph passes its logic tests during construction, perhaps importantly after being expanded for the given domain coverage, its principal use is as a read-only knowledge structure for making subset selections or querying. The standard SPARQL query language, occasionally supplemented by rule-based queries using SWRL or for bulk actions using the OWL API, are the means by which we access the knowledge graph in real time. In many instances, such as for the KBpedia knowledge graph, these are patterned queries. In such instances, we substitute variables in the queries and pass those from the HTML to query templates.

When doing machine learning, generally slices get retrieved via query and then staged for the learner. A similar approach is taken to generate entity lists for things like training recognizers and taggers. Some of the actions may also do graph traversals in order to retrieve the applicable subset.

However, the main real-time use of the knowledge structure is search. This relies totally on SPARQL. We discuss some options on how this is controlled below.

## Hyponymy, Subsumption and Natural Classes


The principal reasoning basis in the knowledge graph is based on hierarchical, hyponymous relations and instance types. These establish the parent-child lineages, and enable individuals (or instances, which might be entities or events) to be related to their natural kinds, or types. Entities belong to types that share certain defining essences and shared descriptive attributes.

For inferencing to be effective, it is important to try to classify entities into the most natural kinds possible. I have spoken elsewhere about this topic [2]; clean classing into appropriate types is one way to

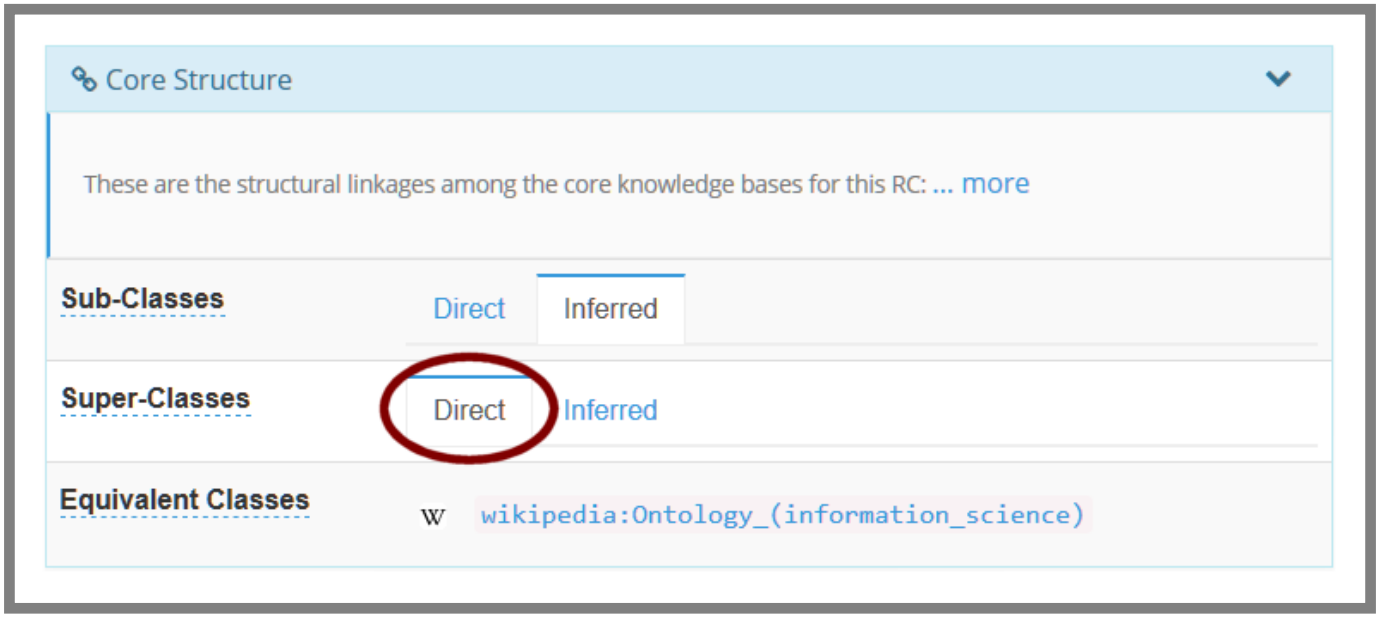
ensure the benefits from related search and related querying are realized. Types may also have parental types in a hyponymous relation. This 'accordion-like' design is an important aspect that enables external schema to be tied into multiple points in KBpedia [3].

Disjointedness assertions, where two classes are logically distinct, and other relatedness options provide other powerful bases for winnowing potential candidates and testing placements and assignments. Each of these factors also may be used in SPARQL queries.

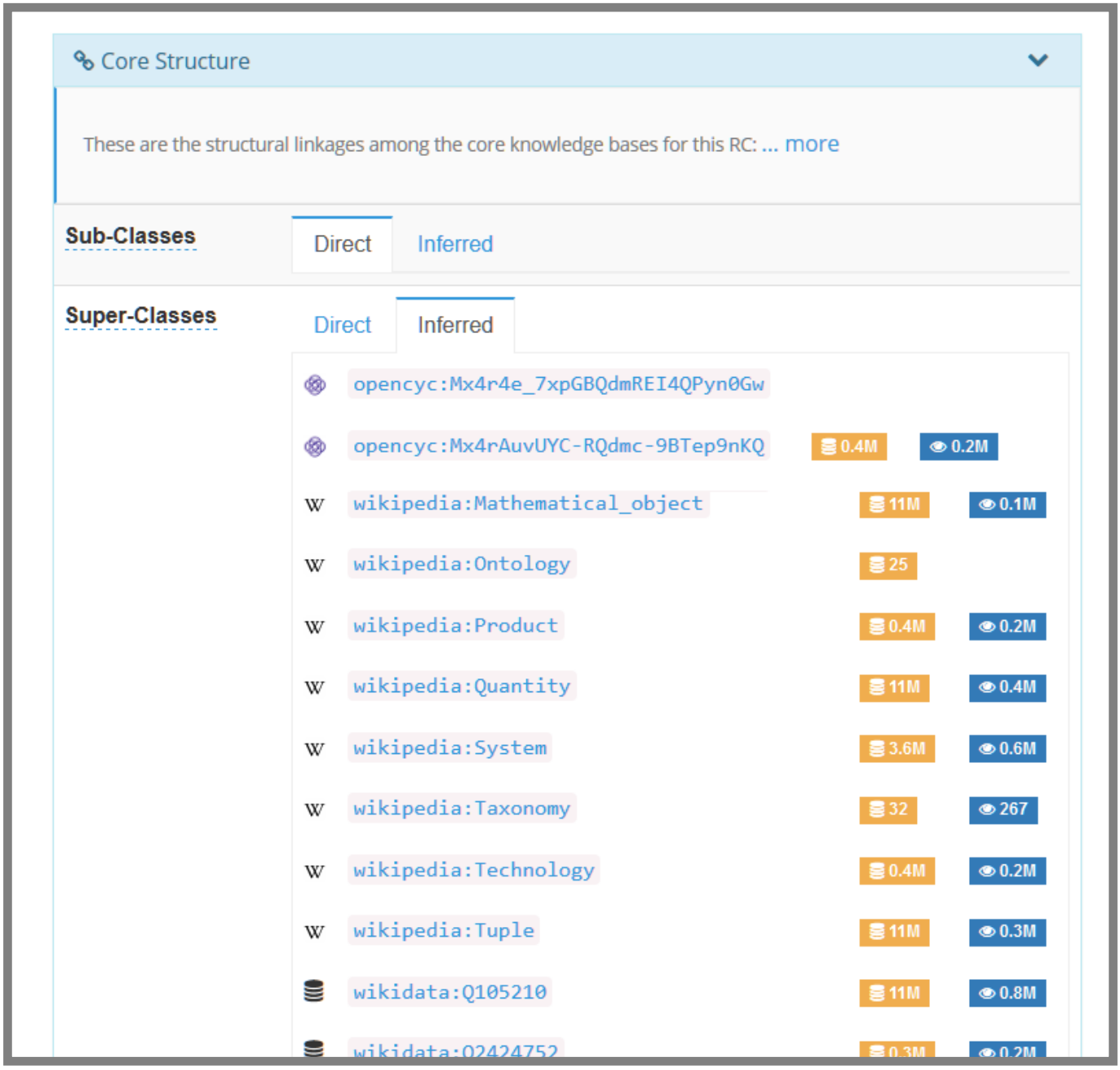
These constructs of semantic Web standards, combined with a properly constructed knowledge graph and the use of synonymous and related vocabularies in *semsets* as described in a previous use case, provide powerful mechanisms for how to query a knowledge base. By using these techniques, we may dial-in or broaden our queries, much in the same way that we choose different types of sprays for our garden watering hose. We can focus our queries to the particular need at hand. We explain some of these techniques in the next sections.

## Adjusting Query Focus

We can see a crude application of this control when browsing the KBpedia knowledge graph. When we enter a particular query, in this case, 'knowledge graph', one result entry is for the concept of ontology in information science. We see that a direct query gives us a single answer:




However, by picking the inferred option, we now see a listing of some 83 super classes for our ontology concept:


Core Structure
These are the structural linkages among the core knowledge bases for this RC: ... more
Sub-Classes
Direct
Inferred
Super-Classes
Direct
Inferred
opencyc:Mx4r4e_7xpGBQdmREI4QPyn0Gw
opencyc:Mx4rAuvUYC-RQdmc-9BTep9nKQ
0.4M
0.2M
wikipedia:Mathematical_object
11M
0.1M
wikipedia:Ontology
25
wikipedia:Product
0.4M
0.2M
wikipedia:Quantity
11M
0.4M
wikipedia:System
3.6M
0.6M
wikipedia:Taxonomy
32
267
wikipedia:Technology
0.4M
0.2M
wikipedia:Tuple
11M
0.3M
wikidata:Q105210
11M
0.8M

By reasoning for deductive inference, we are actually broadening our query to include all of the parental links in the subsumption chain within the graph. Ultimately, this inference chain traces upward into the highest order concept in the graph, namely `owl:Thing`. (By convention, `owl:Thing` itself is excluded from these inferred results.)

By invoking inference in this case, while we have indeed broadened the query, it also is quite indiscriminate. We are reaching all of the ancestors to our subject concept, reaching all of the way to the root of the graph. This broadening is perhaps more than what we actually seek.

## Scoping Queries via Proerty Paths

Among many other options, SPARQL also gives us the ability to query specific property paths [4]. We can invoke these options either in our query templates or programmatically in order to control the breadth and depth of our desired query results.

Let's first begin with the SPARQL query that uses 'knowledge graph' in its `altLabel`:

```
==============
select ?s ?p ?o
from <http://kbpedia.org/1.40/>
where
{
  ?s
<http://www.w3.org/2004/02/skos/core#altLabel> "Knowledge graph"@en ;
     ?p ?o .
}
==============
```

You can see from the results below that only the concept of ontology (information science) is returned as a `prefLabel` result, with the concept's other `altLabels` also shown:

```
==============
s     p     o

http://kbpedia.org/kko/rc/OntologyInformationScience
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/2002/07/owl#Class
http://kbpedia.org/kko/rc/OntologyInformationScience
http://www.w3.org/2000/01/rdf-schema#isDefinedBy
http://kbpedia.org/kko/rc/
```

http://kbpedia.org/kko/rc/OntologyInformationScience
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://kbpedia.org/kko/rc/KnowledgeRepresentation-CW
http://kbpedia.org/kko/rc/OntologyInformationScience
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://kbpedia.org/kko/rc/Ontology
http://kbpedia.org/kko/rc/OntologyInformationScience
http://www.w3.org/2000/01/rdf-schema#subClassOf
http://wikipedia.org/wiki/Ontology (information science)
<http://wikipedia.org/wiki/Ontology %28information science%29>
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02
/skos/core#prefLabel "Ontology (information science)"@en
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/
skos/core#altLabel "Ontological distinction (computer science)"@en
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/
skos/core#altLabel "Ontological distinction(computer science)"@en
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/skos/core#altLabel "Ontology Language"@en
http://kbpedia.org/kko/rc/OntologyInformationScience
http://www.w3.org/2004/02/skos/core#altLabel "Ontology media"@en
http://kbpedia.org/kko/rc/OntologyInformationScience
http://www.w3.org/2004/02/skos/core#altLabel "Ontologies"@en
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/skos/core#altLabel "New media relations"@en
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/skos/core#altLabel "Strong ontology"@en
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/
skos/core#altLabel "Ontologies (computer science)"@en
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/
skos/core#altLabel "Ontology library (information science)"@en
http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/

```
skos/core#altLabel "Ontology Libraries (computer science)"@en
      http://kbpedia.org/kko/rc/OntologyInformationScience
      http://www.w3.org/2004/02/skos/core#altLabel "Ontologing"@en
      http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/
skos/core#altLabel "Computational ontology"@en
      http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/
skos/core#altLabel "Ontology (computer science)"@en
      http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/
skos/core#altLabel "Ontology library (computer science)"@en
      http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/skos/core#altLabel "Populated ontology"@en
      http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/skos/core#altLabel "Knowledge graph"@en
      http://kbpedia.org/kko/rc/OntologyInformationScience

http://www.w3.org/2004/02/skos/core#altLabel "Domain ontology"@en
      http://kbpedia.org/kko/rc/OntologyInformationScience
      http://www.w3.org/2004/02/skos/core#definition
      "In computer science and information science, an ontology is a f
ormal
      naming and definition of the types, properties, and interrelatio
nships of
      the entities that really or fundamentally exist for a particular
 domain
      of discourse."@en
      http://kbpedia.org/kko/rc/OntologyInformationScience
      http://kbpedia.org/ontologies/kko#superClassOf
      http://wikipedia.org/wiki/Ontology_(information_science)
      <http://wikipedia.org/wiki/Ontology_%28information_science%29>
      ==============
```

This result gives us the basis for now asking for the direct parents of our ontology concept, using this query:

```
      ==============
```

```
select ?directParent
from <http://kbpedia.org/1.40/>
where
{
  <http://kbpedia.org/kko/rc/OntologyInformationScience>
  <http://www.w3.org/2000/01/rdf-schema#subClassOf>
  ?directParent .
}
==============
```

We see that the general concepts of *knowledge representation-CW* and *ontology* are parents to our concept, as well as the external Wikipedia result on ontology (information science):

```
==============
directParent

http://kbpedia.org/kko/rc/KnowledgeRepresentation-CW
http://kbpedia.org/kko/rc/Ontology
http://wikipedia.org/wiki/Ontology_(information_science)
<http://wikipedia.org/wiki/Ontology_%28information_science%29>

==============
```

If we turn on the inferred option, we will get the full listing of the 83 concepts noted earlier. This is way too general for our current needs.

While it is not possible to specify a depth using SPARQL, it is possible to use property paths to control the extent of the query results from the source. In this case, we specify a path length of 1:

```
==============
select ?inferredParent
from <http://kbpedia.org/1.40/>
where
{
  <http://kbpedia.org/kko/rc/OntologyInformationScience>
  <http://www.w3.org/2000/01/rdf-schema#subClassOf>{,1}
  ?inferredParent .
}
==============
```

Which produces results equivalent to the "direct" search (namely, direct parents only):

```
==============
directParent

http://kbpedia.org/kko/rc/KnowledgeRepresentation-CW
http://kbpedia.org/kko/rc/Ontology
http://wikipedia.org/wiki/Ontology_(information_science)
<http://wikipedia.org/wiki/Ontology_%28information_science%29>
==============
```

However, by expanding our path length to two, we now can request the parents and grandparents for the ontology (information science) concept:

```
==============
select ?inferredParent
from <http://kbpedia.org/1.40/>
where
{
  <http://kbpedia.org/kko/rc/OntologyInformationScience>
    <http://www.w3.org/2000/01/rdf-schema#subClassOf>{,2}
  ?inferredParent .
}
=============
```

This now gives us 15 results from the parental chain:

```
==============
inferredParent

http://kbpedia.org/kko/rc/OntologyInformationScience
http://wikipedia.org/wiki/Ontology_(information_science)
<http://wikipedia.org/wiki/Ontology_%28information_science%29>
http://kbpedia.org/kko/rc/Ontology
http://kbpedia.org/kko/rc/KnowledgeRepresentation-CW
http://umbel.org/umbel/rc/KnowledgeRepresentation-CW
http://kbpedia.org/kko/rc/PropositionalConceptualWork
http://wikipedia.org/wiki/Knowledge_representation
http://sw.opencyc.org/concept/Mx4r4e_7xpGBQdmREI4QPyn0Gw
http://umbel.org/umbel/rc/Ontology
http://kbpedia.org/kko/rc/StructuredInformationSource
```

```
http://kbpedia.org/kko/rc/ClassificationSystem
http://wikipedia.org/wiki/Ontology
http://sw.opencyc.org/concept/Mx4rv7D_EBSHQdiLMuoH7dC2KQ
http://kbpedia.org/kko/rc/Technology-Artifact
http://www.wikidata.org/entity/Q324254

==============
```

Similarly we can expand our query request to a path length of 3, which gives us the parental chain from parents + grandparents + great-grandparents):

```
==============
select ?inferredParent
from <http://kbpedia.org/1.40/>
where
{
  <http://kbpedia.org/kko/rc/OntologyInformationScience>
     <http://www.w3.org/2000/01/rdf-schema#subClassOf>{,3}
  ?inferredParent .
}
=============
```

In this particular case, we do not add any further results for great-grandparents:

```
==============

inferredParent

http://kbpedia.org/kko/rc/OntologyInformationScience
http://wikipedia.org/wiki/Ontology_(information_science)
<http://wikipedia.org/wiki/Ontology_%28information_science%29>
http://kbpedia.org/kko/rc/Ontology
http://kbpedia.org/kko/rc/KnowledgeRepresentation-CW
http://umbel.org/umbel/rc/KnowledgeRepresentation-CW
http://kbpedia.org/kko/rc/PropositionalConceptualWork
http://wikipedia.org/wiki/Knowledge_representation
http://sw.opencyc.org/concept/Mx4r4e_7xpGBQdmREI4QPyn0Gw
http://umbel.org/umbel/rc/Ontology
http://kbpedia.org/kko/rc/StructuredInformationSource
http://kbpedia.org/kko/rc/ClassificationSystem
http://wikipedia.org/wiki/Ontology
```

```
http://sw.opencyc.org/concept/Mx4rv7D_EBSHQdiLMuoH7dC2KQ
http://kbpedia.org/kko/rc/Technology-Artifact
http://www.wikidata.org/entity/Q324254
==============
```

Without a property path specification, our inferred request would produce the listing of 83 results shown by the Inferred tab on the KBpedia knowledge graph, as shown in the screen capture provided earlier.

The online knowledge graph does not use these property path restrictions in its standard query templates. But these examples show how programmatically it is possible to broaden or narrow our searches of the graph, depending on the relation chosen (`subClassOf` in this example) and the length of the specified property path.

## Many More Options and Potential for Control

This use case is but a small example of the ways in which SPARQL may be used to dial-in or control the scope of queries posed to the knowledge graph. Besides all of the standard query options provided by the SPARQL standard, we may also remove duplicates, identify negated items, and search inverses, selected named graphs or selected graph patterns.

Beyond SPARQL and now using SWRL, we may also apply abductive reasoning and hypothesis generation to our graphs, as well as mimic the action of expert systems in AI through if-then rule constructs based on any structure within the knowledge graph. A nice tutorial with examples that helps highlight some of the possibilities in combining OWL 2 with SWRL is provided by [5].

A key use of inference is its ability to be applied to natural language understanding and the extension of our data systems to include unstructured text, as well as structured data. For this potential to be fully realized, it is important that we chunk ("parse") our natural language using primitives that themselves are built upon logical foundations. Charles S. Peirce made many contributions in this area as well. Semantic grammars that tie directly into logic tests and reasoning would be a powerful addition to our standard semantic technologies. Revisions to the approach taken to Montague grammars may be one way to achieve this illusive aim. This is a topic we will likely return to in the months to come.

Finally, of course, inference is a critical method for testing the logic and consistency of our knowledge graphs as we add new concepts, make new relations or connections, or add attribute data to our instances. All of these changes need to be tested for consistency moving forward. Nurturing graphs by testing added concepts, entities and connections is an essential prerequisite to leveraging inferencing at run time as well.

This article is part of an occasional series describing non-machine learning use cases and applications for Cognonto's KBpedia knowledge graph. Most center around the general use and benefits of knowledge graphs, but best practices and other applications are also discussed. Prior machine learning use cases, and the ones from this series, may be found on the Cognonto Web site under the Use Cases main menu item.

[1] See, for example, Markus Krötzsch, Frantisek Simancik, and Ian Horrocks, 2012. "A Description Logic Primer." arXiv preprint, *arXiv*:1201.4089; and Franz Baader, 2009. "Description Logics," in Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Reasoning Web. Semantic Technologies for Information Systems – 5th International Summer School*, 2009, volume 5689 of LNCS, pages 1–39. Springer, 2009.

[2] M.K. Bergman, 2015. "'Natural Classes' in the Knowledge Web," in *AI3:::Adaptive Information* blog, July 13, 2015.

[3] M.K. Bergman, 2016. "Rationales for Typology Designs in Knowledge Bases," in *AI3:::Adaptive Information* blog, June 6, 2016.

[4] Steve Harris and Andy Seaborne, eds., 2013. SPARQL 1.1 Query Language, *World Wide Web Consortium (W3C) Recommendation*, 21 March 2013; see especially Section 9 on property paths.

[5] Martin Kuba, 2012. "Owl 2 and SWRL Tutorial," from Kuba's Web site.

---