Open Information Systems 2019-2020

Lecture 8: Evaluating and Validating Ontologies and Knowledge Bases

Christophe Debruyne

Ontologies vs. Knowledge Bases

- Ontologies are a [formal,] explicit specification of a [shared] conceptualization [Gru95] and extended by [Stu98]

- Some argue there is a difference between ontologies and knowledge bases (e.g., Hepp 2008):
  - The ontology corresponds with the "schema" – the TBox
  - The knowledge base corresponds with the ontology and instances – TBox and ABox together

- What are the advantages of considering the two separately?
- As always, there are "gray areas" where it makes sense to include some individuals in the ontology. Hepp proposes to differentiate between *ontology individuals* and *data individuals*.
- Can you come up with examples?

Context

While ontologies and KBs are different, we will – in this lecture – use the term "ontology" for both unless otherwise explicitly specified.

- Ontologies (and knowledge bases) are built; they have to be *engineered*. Therefore they need to be *evaluated*.

- Ontologies – however – have some unique challenges
  - Unlike software, ontologies cannot be compiled and run.
  - Ontologies can be used for different tasks.
  - Ontologies can be even used in unforeseen ways.
  - Ontologies can be used for the integration and reuse of heterogeneous data sources, which require mappings and the resulting KBs may produce different results.

Ontology Evaluation

- We distinguish between verification and validation:
  - Ontology verification aims to answer the question: "Was the ontology built in the right way?"
  - Ontology validation aims to answer the question: "Was the right ontology built?"
- Can you come up with examples of assessing both aspects?
- Some dimensions – such as the correct implementation of ontology requirements – can be considered part of both verification and evaluation.
- While the latter is mostly outside the scope of this lecture, ontology validation may require techniques such as human assessment, technical action research, certification, etc.

Ontology Evaluation

- Ontology evaluation is thus checking the technical quality of an ontology (or knowledge base) against a frame of reference and is a critical activity in ontology engineering projects (Poveda-Villalón, M et al., 2014).

Ontology Quality Criteria

Accuracy	Adaptability
Clarity	Completeness
Computational Efficiency	Conciseness
Consistency	Organizational Fitness

Ontology Quality Criteria

- The criteria provide a framework for assessing the ontology.
- But some criteria are even contradictory (!)
  - **Completeness vs. minimal ontological commitment.**
- Criteria have to be assessed and evaluated. Certain queries might take a long while to compute (computational efficiency), but only asked occasionally and not time-critical (organizational fitness).
- The requirements of your ontology project may even affect the criteria: e.g., the choosing to remain within OWL 2 QL will have an impact on some of the reasoning tasks.
- You, as an evaluator, is to first choose what criteria are relevant and then decide how to assess how well your ontology meets those criteria. Criteria provide justifications for evaluation methods.

Ontology Quality Criteria

- *The following "criteria" is merely a way to classify aspects that can be assessed. The goal of this list is to provide a framework.*

- **Accuracy**
  - Does the ontology comply with reality?
  - Does the ontology capture the knowledge of domain experts (and users)?
  - Truth vs. Consensus (Hepp 2008)
- **Adaptability**
  - Does the ontology anticipate its uses?
  - Does the ontology provide the foundation for multiple tasks?
  - Does it allow extension, integration and adaptation in a monotonic way?
    - I.e., Can we extend an ontology without removing axioms?
  - Why do you think monotonicity is important?
  - How can you facilitate adaptability in RDFS and OWL, and how would you implement it?
  - Usability vs. Reuse (Spyns et al. ,2002)

Ontology Quality Criteria

- **Clarity**
  - Does the ontology effectively communicate the intended meaning of concepts, relations, and instances?
  - Are the definitions documented (labels, comments, references, …)?
- **Completeness**
  - Is the UoD appropriately covered?
  - Does the ontology meet the requirements?
  - Cfr. Competency questions (Gruninger and Fox, 1995).
    - Competency questions are questions that the ontology should be able to (help) to answer. The answer can be sometimes found by the ontology (via reasoning), the query language (SPARQL), or by an application on top of of those.
    - When nor the ontology or the query language can provide the answer, it is advised the reformulate the competency question. E.g., instead of “computing trends” the ontology should “provide all necessary information for computing trends”.

Ontology Quality Criteria

- **Computational efficiency**
  - How easily / successfully can reasoners process the ontology?
  - How efficient are certain reasoning tasks?
  - Note: Remember the various OWL profiles: EL, RL, and QL.
- **Conciseness**
  - Are there irrelevant or redundant axioms?
    - What is the difference between irrelevant and redundant?
    - Is redundancy always bad? Discuss?
  - Does only contain only the essential terms?
    - Also called minimal ontological commitment (Gruber, 1995)

Example from the GLAM sector:
- Dublin Core: -> names of people are literals
- CIDOC CRM: names are instances of *Appellation* identifying instances of *Person,* and relationships between appellations can be declared.

Ontology Quality Criteria

- **<u>Consistency and Coherence</u>**
  - Are there logical contradictions?
  - Do formal and informal definitions of concepts match?
  - Minimal encoding bias – conceptualize at the "knowledge level" and conceptualization should not depend on the encoding. – i.e., do not let the ontology language or encoding drive ontology engineering!
    - "Dirty hacks" for convenience?
    - Lists and containers vs. ontology design patterns
- **<u>Organizational Fitness</u>**
  - Used by the stakeholders? E.g., an organization
  - Adequate method for ontology construction adopted?
  - Is it properly shared among stakeholders?
  - Does it comply with legislation?
    - E.g., GDPR.

Ontology Aspects

- Criteria are difficult to ‘measure’ automatically. However, some ontology *aspects* can be – to some extent – analyzed automatically.

Vocabulary	Syntax
Structure	Semantics
Representation	Context

- But, there are degrees of freedom that needs to be taken into account, and that freedom depends on the type of ontology project:
  - E.g., the use of HTTP URIs is important for Linked Data.

Ontology Aspects: Vocabulary

- An ontology's vocabulary is the set of all names of an ontology's terms: URIs and literals.
- URIs:
  - URLs, URIs, URNs – *When to use which?*
  - Opaque vs. readable URIs
  - Reusing URIs (and monitor the evolution of the reused resource)
  - Ontology languages allow one to infer information about URIs, with `ex:Foo a ex:Bar`, a reasoner can infer that Foo is an individual and Bar a class.
- Literal:
  - Labels should exist for all URIs, according to particular style guidelines
    - Helps assessing typos, for instance
  - Typed literals can be used to validate values

- **What can we "automate"? Typos can be checked (ex:Address vs. ex:Adress), documentation can be checked, style guides can be checked, and types of the literals can be checked,…**

Ontology Aspects: Syntax

- Describing graph returning ontology (RDF/XML, NTriples)
- Describing ontology directly and can be transformed into graph (OWL Manchester Syntax)
- Can be transformed from one into the other, but things might get lost, e.g., XML comments in RDF/XML. Avoid such comments and use ontology annotation instead.

- Things that can be checked:
  - Proper indentation
  - Ordering of the information (axioms before facts, grouping around individuals, etc.)
  - Simple RDF/XML KBs can be accompanied by additional DTDs or XML Schemas.

Ontology Aspects: Structure

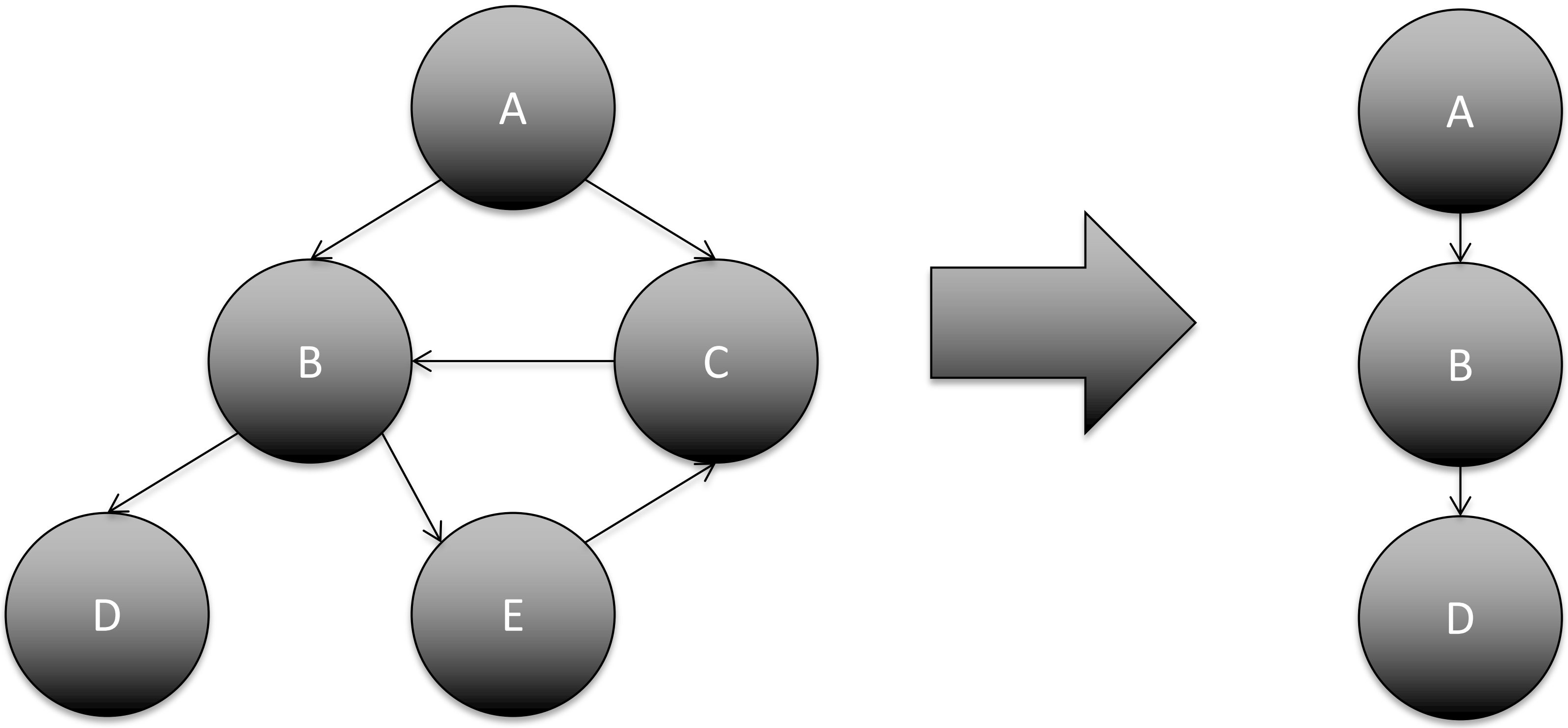
- Investigating <u>the structural properties of the graph</u>
  - Number of classes
  - Number of properties
  - Complexity of trees
  - Where are disjointness axioms declared?
  - Punning
  - “Anti-patterns”
- Are there circularities in hierarchies? This may indicate an error (related to semantics, see later)
- Are classes used as individuals? This may indicate a problem unless punning was intended (also related to semantics)
- <u>These metrics are easy to compute, but are merely indicators.</u>

Ontology Aspects: Semantics

- Can the ontology be **normalized** (Vrandecic and Sure, 2007). Normalization is making certain features explicit while retaining semantics. This is for example achieved by:
  - Naming all classes and individuals – no anonymous (complex) classes and individuals
  - Compute transitive closure and normalize names (so that classes with more than one name only have one), etc.
  - Use deepest possible class or relations only:
    - :Cat(:gaston), ~~:Pet(:gaston)~~, rdfs:subClassOf(:Cat, :Pet)
  - Materialize reflexive, symmetrical, inverse properties and simplify the transitivity graphs.
- Benefits? No cycles in ontology, number of names and classes will correspond, etc. Normalization makes it easier to define metrics.

Ontology Aspects: Semantics

Example of normalization from Vrandecic 2009.



What is wrong here?

Ontology Aspects: Semantics

- Stability of the ontology
  - Can the taxonomy collapse by adding axioms, which may indicate a weak taxonomy.
  - Are classes "closed" by providing an exhaustive enumeration of individuals that together form the instances of a class?

  -
    ```
    EquivalentClasses( ex:PlanetInOurSolarSystem
      ObjectOneOf( ex:mercury, ex:venus, ex:earth, ex:mars,
                   ex:jupiter, ex:saturn, ex:uranus, ex:neptune ) )
    ```

Ontology Aspects: Representation

- Relation semantics and structure.
- Analysis of semantics and structure can indicate possible problems.
- The example before had a different structure, but same semantics
  - Is this an error? Was it done intentionally? The more concise ontology seems more favorable, yet the first could have a valid reason. That reason, however, needs to be documented.

Ontology Aspects: Context

- Evaluate ontology with additional formalized artifacts

- Example 1. Competency questions (Gruninger and Fox 1995) → formulate in adequate query language. Can ontology help one to answer that query?
- Example 2. Unit tests for ontology.
  – Protégé includes support for unit tests
- Example 3. Checking for additional constraints.
  – E.g., every person must have name. How would one model this?
  – Cfr. Jiao *et al.* (2010) – for instance.

OOPS!

- OntOlogy Pitfall Scanner! (http://oops.linkeddata.es/)

- Poveda-Villalón, M., Gómez-Pérez, A., & Suárez-Figueroa, M. C. (2014). OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS), 10*(2), 7-34. doi:10.4018/ijswis.2014040102

- Authors analyzed hundreds of ontologies to identify various "pitfalls" and created a service that assesses one's ontology w.r.t. their analysis.

- Again, problems picked up by OOPS! need to be investigated; it may be that some problems were intentional.

Knowledge Base Verification

- Open World Assumption: unknown information is not necessarily false. But what if we want to check whether our knowledgebase contains all necessary triples. In that case, we have to assess the graph rather than the ontology/knowledgebase.

- One such attempt was proposed by T., Jiao, E. Sirin, J. Bao, and D. L. McGuinness. Integrity Constraints in OWL. In AAAI. 2010. They "treated" OWL axioms as integrity constraints by translating those into a set of SPARQL ASK questions.

Knowledge Base Verification

- T., Jiao, E. Sirin, J. Bao, and D. L. McGuinness. Integrity Constraints in OWL. In AAAI. 2010, for instance, translates constraints into a set of SPARQL ASK questions as SPARQL queries the graph.
- Every product is produced by a producer
$$Product \sqsubseteq \exists hasProducer.Producer$$
- Then we assert $Product(carrot)$
- With the axiom above, we know that carrot is produced by at least one unknown producer. But what if we want to use that axiom to test whether our KR has a triple stating such a relationship?
- Jiao proposed a framework that “translates” axioms into SPARQL ASK queries to use these as “integrity constraints”:
- ASK WHERE {
 ?x rdf:type ex:Product.
 NOT EXISTS{ ?x ex:hasProducer ?y. ?y rdf:type ex:Producer. } }

Knowledge Base Verification

- Now there are initiatives to standardize knowledge base verification.
- Shapes Constraint Language (SHACL), a W3C Recommendation from 2017 https://www.w3.org/TR/shacl/
- ShEx, or Shape Expressions, currently under active development https://shex.io/ (a community effort)

- You declare how pieces of the graph should look look like; i.e., you declare valid shapes in the graph.

- We will consider a simple example. The SHACL implementation I use is provided by topbraid, currently the only freely available implementation that is fully compliant with the Recommendation.

SHACL Example: our graph

```
@base <http://example.org/>
@prefix ex: <http://example.org/ontology#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<#circle1>
   a ex:Circle ;
   ex:radius "4"^^xsd:integer
   .

<#circle2>
   a ex:Circle ;
   ex:radius "1"^^xsd:integer
   .
```

SHACL Example: SHACL Constraints

```
@base <http://example.org/>
@prefix ex: <http://example.org/ontology#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix sh:   <http://www.w3.org/ns/shacl#> .

ex:CircumferenceShape
   a sh:NodeShape ;
   sh:targetClass ex:Circle ;

   sh:property [
      sh:path ex:radius ;
      sh:minCount 1 ;
      sh:maxCount 1 ;
   ] ;
   .
```

All Circles must have exactly one radius.

SHACL Example: SHACL Constraints

```java
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.Resource;
import org.topbraid.shacl.rules.RuleUtil;
import org.topbraid.shacl.validation.ValidationUtil;

public class ShaclTest1 {
    public static void main(String[] args) {
        Model data = ModelFactory.createDefaultModel();
        data.read("./resources/test.ttl");
        Model shapes = ModelFactory.createDefaultModel();
        shapes.read("./resources/shapes.ttl");
        Resource report = ValidationUtil.validateModel(data, shapes, true);
        report.getModel().write(System.out, "TURTLE");
    }
}
```

```
[ a       <http://www.w3.org/ns/shacl#ValidationReport> ;
  <http://www.w3.org/ns/shacl#conforms>
          true
] .
```

SHACL Example: SHACL Rules

```
@base <http://example.org/>
@prefix ex: <http://example.org/ontology#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix sh:   <http://www.w3.org/ns/shacl#> .

ex:CircumferenceShape
   a sh:NodeShape ;
   sh:targetClass ex:Circle ;

   sh:rule [
      a sh:JSRule ;
      sh:jsFunctionName "circumference" ;
      sh:jsLibrary [
         sh:jsLibraryURL "file:///.../circumference.js"^^xsd:anyURI
      ] ;
   ] ;   .
```

SHACL Example: circumference.js

```
var NS = "http://example.org/ontology#";

function circumference($this) {
    var radius = getProperty($this, "radius");
    var circumference = TermFactory.literal(radius.lex * 2 * 3.14, radius.datatype);
    // Create a triple. Note: it is important to use $this for the subject :-)
    return [ [$this, TermFactory.namedNode(NS + "circumference"), circumference ] ] ;
}

function area($radius) {
    return radius.lex * radius.lex * 3.14;
}

function getProperty($this, name) {
    var it = $data.find($this, TermFactory.namedNode(NS + name), null);
    var result = it.next().object;
    it.close();
    return result;
}
```

SHACL Example: SHACL Constraints

```java
public class ShaclTest2 {
    public static void main(String[] args) {
        Model data = ModelFactory.createDefaultModel();
        data.read("./resources/test.ttl");
        Model function = ModelFactory.createDefaultModel();
        function.read("./resources/function.ttl");
        Model inferenced = RuleUtil.executeRules(data, function, null, null);
        inferenced.write(System.out, "TURTLE");    }
}
```

```turtle
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

<http://example.org/#circle2>
        <http://example.org/ontology#circumference>
                "6.28"^^xsd:integer .

<http://example.org/#circle1>
        <http://example.org/ontology#circumference>
                "25.12"^^xsd:integer .
```

SHACL Example: SHACL Constraints

```
@base <http://example.org/>
@prefix ex: <http://example.org/ontology#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix sh:   <http://www.w3.org/ns/shacl#> .

ex:CircumferenceShape
   a sh:NodeShape ;
   sh:targetClass ex:Circle ;

   sh:property [
      sh:path ex:radius ;
      sh:minCount 1 ;
      sh:maxCount 1 ;
      sh:severity sh:Warning ;
   ] ;
   .
```

You can declare the severity of certain shapes: Info, Warning, and Violation. Violation is the default severity.

Conclusions

- Ontology and Knowledge Base **Evaluation** and **Validation**
- There are certain criteria that we can assess. We have covered about 8 of these criteria.
- Criteria are difficult to measure in a automatic way, but we can look at aspects that are related to these criteria. The quality criteria clarity can be assessed by looking into vocabulary aspects, for instance.
- OWL Reasoners can be used to assess certain aspects.
- We finally saw the use of OWL and SHACL as a way to validate knowledge bases.

References

- Vrandečić, Denny. "Ontology evaluation." Handbook on ontologies. Springer, Berlin, Heidelberg, 2009. 293-313.
- Brewster, Christopher, et al. "Data driven ontology evaluation." Proceedings of he International Conference on Language Resources and Evaluation, Portugal. 24 - 30 May 2004, (2004).
- Brank, Janez, Marko Grobelnik, and Dunja Mladenic. "A survey of ontology evaluation techniques." Proceedings of the conference on data mining and data warehouses (SiKDD 2005). Citeseer Ljubljana, Slovenia, 2005.
- T. Gruber. Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum.-Comput. Stud., 43(5-6):907–928, 1995.
- R. Studer, R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. Data & Knowledge Engineering, 25(1–2):161–198, 1998.
- Hepp, Martin. "Ontologies: State of the art, business potential, and grand challenges." Ontology Management. Springer, Boston, MA, 2008. 3-22.
- Peter Spyns, Robert Meersman, Mustafa Jarrar: Data Modelling versus Ontology Engineering. SIGMOD Record 31(4): 12-17 (2002)
- M. Gruninger and M. S. Fox. Methodology for the design and evaluation of ontologies. In IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, 1995.
- D. Vrandecic and A. Gangemi. Unit tests for ontologies. In M. Jarrar, C. Ostyn, W. Ceusters and A. Persidis, editors, Proceedings of the 1st International Workshop on Ontology Content and Evaluation in Enterprise, LNCS, Montpellier, France. Springer, Berlin, 2006.
- D. Vrandecic and Y. Sure. How to design better ontology metrics. In W. May and M. Kifer, editors, Proceedings of the 4th European Semantic Web Conference (ESWC 2007), Innsbruck, Austria. Springer, Berlin, 2007.
- D. Vrandecic. Ontology evaluation. In S. Staab and R. Studer, editors, Handbook on Ontologies in Information Systems, Second Edition, Internal Handbooks on Information Systems, pages 293-313. Springer, Berlin, 2009.
- T., Jiao, E. Sirin, J. Bao, and D. L. McGuinness. Integrity Constraints in OWL. In AAAI. 2010.