Challenges in the Core of Ontology Support Systems

Peter F. Patel-Schneider

Nuance Communications
pfpschneider@gmail.com

6 October 2012

Abstract

The effective use of ontology languages, such as the W3C Semantic Web Languages OWL and RDFS, requires a complex support system, with developer interface, development cycle, knowledge acquisition, learning, application interface, and data access components as well as the core components that actually implement reasoning and querying using the ontology language. Usable versions of these core components exist for RDFS and the various versions of OWL that can reason with many large complex ontologies, and integrated systems that support the use of OWL and similar languages are now proliferating. However, the current reasoners are less capable when handling large amounts of data and expressive ontologies, and there remain daunting challenges in building reasoners supporting the full use of ontology languages.

Table of Contents

Table of Contents

Ontologies (KR version)

- Ontologies model some portion of the world,

  E.g., anatomy, diseases, aircraft engines, cell phones, social networks, academic publishing.

- Ontologies introduce a vocabulary,

  E.g., organs, muscles, circulatory system, heart, diseases.

- Ontologies specify meaning of vocabulary ,

  E.g., a person has a name, an age, . . . ,
  E.g., the heart is a muscular organ in the circulatory sytem,
  E.g., heart disease is precisely a disease of the heart.

- All in a formal language (a logic).

Sample Ontology Fragment

- The heart is a muscular part of the circulatory system.

  $Heart \sqsubseteq MuscularOrgan \sqcap \exists isPartOfCirculatorySystem$
- Heart disease is precisely a disease of the heart.

  $HeartDisease \equiv Disease \sqcap \exists affects.Heart$
- Circulatory disease is precisely a disease that affects of a part of the circulatory system.

  $CirculatoryDisease \equiv$
  $Disease \sqcap \exists affects.(\exists isPartOf.CirculatorySystem)$

Without Ontologies

- Given data that Alex has heart disease:

  $\texttt{Alex hasDisease d1}$

  $\texttt{d1} \in \texttt{HeartDisease}$

- Ask who has circulatory disease?

  $?who \in \exists \texttt{hasDisease.CirculatoryDisease}$

- No one can be determined to have a circulatory disease.
  - Because no connection in the data between heart disease and circulatory disease.

With Ontologies

- Given data that Alex has heart disease:

  $\texttt{Alex hasDisease d1}$

  $\texttt{d1} \in \texttt{HeartDisease}$

- . . . and an ontology about anatomy and diseases, including

  $\texttt{Heart} \sqsubseteq \texttt{MuscularOrgan} \sqcap \exists \texttt{isPartOf.CirculatorySystem}$

  $\texttt{HeartDisease} \equiv \texttt{Disease} \sqcap \exists \texttt{affects.Heart}$

  $\texttt{CirculatoryDisease} \equiv$

  $\texttt{Disease} \sqcap \exists \texttt{affects.}(\exists \texttt{isPartOf.CirculatorySystem})$

- Who has circulatory disease?

  Ask: $?who \in \exists \texttt{hasDisease.CirculatoryDisease}$

  Answer: Alex has circulatory disease.

- Is heart disease a specialization of circulatory disease?

  Ask: $\texttt{HeartDisease} \sqsubseteq \texttt{CirculatoryDisease}$

  Answer: Yes.

Effects of Using Ontologies

- Ontologies provide . . .
  - a formalization of part of the world,
  - background information (theories),
  - links between different vocabularies.
- Ontologies permit . . .
  - reasoning with both data (facts) and theories,
  - combining multiple sources of information,
  - much richer inference.
- Ontologies support more intuitive representation and reasoning.
  - Separation of facts and background theories.
  - Differing, intertwined vocabularies.
  - Discovery of "implicit" information.

Example Ontology Uses

- BBC Sport
  - Uses data about sporting events from different sources.
  - Ontologies relate different vocabularies (match vs game).
  - Ontologies provide background information.
  - Web application combines views of sporting events and background stories.
- Samsung
  - Ontologies describe sensor data.
  - Ontologies formalize context (location, ...).
  - Reasoners infer context from data.
  - Applications exploit context to augment behavior.
- Nuance
  - Ontologies provide background theories and data for NL systems.
  - Ontologies provide structuring for NL processing.
  - Used heavily in healthcare applications, in particular.

Table of Contents

The Semantic Web

- Tim Berners-Lee's initial vision of the web was much more than just browsers, and included a notion of a web of data with a well-defined meaning
- This vision as come to be known as the Semantic Web, but the "Visual" Web took off much faster and is much better known
- The current description of the Semantic Web is "a web of data that can be processed directly and indirectly by machines"

Making the Semantic Web Work—Data

- Data in the Semantic Web is encoded as RDF triples
- An RDF triple consists of a subject, a predicate, and an object

  E.g., `pfps:Peter foaf:knows ox:Ian .`

  E.g., `pfps:Peter foaf:age "57"^^xsd:int .`

  E.g., `pfps:Peter rdf:type foaf:Person .`
  - Subjects, predicates, and objects are names (URIs or anonymous names) or literals (data values)
- URIs are global names, which can be used elsewhere, emphasizing connections within the Semantic Web
- RDF triples are available in Web documents, emphasizing accessibility in the Semantic Web

Making the Semantic Web Work—Ontologies

- Semantic Web has
  - Different amounts of information in different sources.
  - Variable levels of curation of data in different sources.
  - Vocabulary shifts between sources.
  - Varying specificity levels between sources.
- Ontologies help overcome problems in all the above.
- Ontologies provide machine-processable meanings for (the names in) the data, particularly the categories and relationships.

Ontology systems that can handle large amounts of data are an important part of the Semantic Web.

Languages for Ontologies in the Semantic Web

RDFS—an extremely simple ontology language, with

- classes and properties,
- sub-classes and sub-properties,
- domains and ranges for properties.

OWL—an expressive ontology language based on Description Logics, with

- exact definitions of classes,
- boolean class combinations,
- class-specific restrictions on properties—range and number,
- transitive, symmetric, and reflexive properties,
- inverses of properties,
- nominals,
- ....

Table of Contents

Systems for Supporting Ontology Use in Applications

Need to provide all phases of system development, including

- Ontology design and management
  - Ontology building and alignment
    - Ontology reasoning—classification
  - Collaborative design and revision control
- Environment analysis
  - Extracting ontologies from data
    - Non-standard reasoning—abduction, least common subsumer
  - Finding relevant sources of data
- Application design
  - Application and language interfaces
    - Ontology reasoning—subsumption
  - Information control (loading ontologies and data)
  - Integrity conditions—determining what data is adequate
- Application execution
  - Individual reasoning—instances of classes, querying
  - Hybrid reasoning, with rules, for example
  - Data lifting, e.g., XML to RDF

Existing Ontology Systems and Components

- Development environments / combinations
  - Protégé—Stanford
  - TopBraid Composer—TopQuadrant
  - Altova Semantic Works—Altova
  - Neon toolkit—Karlsruhe, ...
  - Knoodl—Revelytix
  - KAON2—FZI
  - OntoStudio / OntoBroker—Ontoprise
  - OWL API - interface to OWL—Manchester
  - ...
- Reasoners
  - For RDFS++: AllegroGraph, Sesame
  - For OWL QL: Owlgres, Quill, QuOnto
  - For OWL EL: CB, CEL, ELK, REL
  - For OWL RL: Jena, ELLY, Oracle
  - For OWL: Pellet, Hermit, FaCT++, RacerPro, (SHER)

Table of Contents

Core Component of Complete Ontology Systems

Reasoning!

- If you can't do the reasoning (effectively), you can't use the ontology, and you can't build the applications.
- Reasoning with expressive ontologies is hard in general.
  - Definitely intractable in the worst case.
  - But is it hard in practice?
  - Yes—early reasoners often had severe problems with reasoning.
- There are different reasoning tasks—subsumption, classification, instances, realization, querying, ...
  - They are reducible to each other (more or less),

Reasoning Tasks

Subsumption Is one concept more specific than another in an ontology?

- Is heart disease is a kind of circulatory disease?

Classification Determine all the subsumption relationships between the named concepts in an ontology.

Instances Given an ontology and some data, is an individual an instance of a concept?

- Does Alex have circulatory disease?

Realization Given . . . , find all the named concepts that each named individual is an instance of.

Querying Given . . . , find all the bindings for a conjunctive query.

- Find all the pairs of people and their circulatory diseases.

Reasoning

- The reasoning tasks are reducible to each other, in principle.
  - Data and individuals can be turned into special concept defintions.
- How hard is reasoning in theory?
  - Subsumption is 2NExpTime-complete in OWL!
  - But many subsumptions in an ontology are trivial.
- How hard is reasoning in practice?
  - Often studied via difficulty of classification of ontologies.
  - Classification ends up computing many subsumption relationships, so adds up a lot of little reasoning tasks
    - Some trivial, some easy, some hard.

Classification Times for GALEN—mid 1990s

GALEN is an ontology of medical procedures and related information with about 1500 named concepts.

Classifying GALEN (i.e., determining all subsumptions) :

Times in seconds	Kris	Crack
Load	135.90	—
Pre-process	—	—
Classify	$\gg$400,000	$\gg$10,000
Total CPU time	$\gg$400,000	$\gg$10,000

Both Kris and Crack get about 1/3 of the way through and then get stuck on a hard subsumption test.

Tableau Reasoning for Ontologies

Tableau reasoning:

- Is a way of doing reasoning in logics using model-building.
- Can be adapted to do ontology reasoning.
- A tableau encodes multiple models, including choice points.
- Was considered to be slow, compared to other methods (e.g., resolution).
- Aggressive optimizations can make tableau fast.
  - FaCT and DLP systems are highly-optimized tableau reasoners.

Classification Times for GALEN—late 1990s

Classifying GALEN:

Times in seconds	Kris	Crack	FaCT	DLP
Load	135.90	—	6.03	—
Pre-process	—	—	0.85	—
Classify	≫400,000	≫10,000	204.03	—
Total CPU time	≫400,000	≫10,000	210.91	69.56

Most subsumption tests in FaCT and DLP are instantaneous, and the previously-hard ones generally take under one second.
FaCT and DLP work fast because they aggressively drive towards potential solutions and don't do unnecessary work.

Some Optimizations for Tableau Reasoning

- Ask easy questions first—syntactic checks.
- Use answers over and over—propagation in taxonomies.
- Ask questions with large impact first.
- Transform input to eliminate hard constructs.
- Normalize to enable more quick answers.
- Use heuristics to select among good next things to do.
- Stop unnecessary work as soon as possible.
- Do deterministic processing first—BCP.
- Branch on disjunct and its negation, not other disjuncts.
- Remember partial answers during reasoning—caching.
- Make good guesses—effective heuristics.
- Don't redo useless work—backjumping.

Results of Optimizations

DLP Classification times for GALEN (CPU seconds):

Optimisation Removed	Selection Heuristic Used			
	Oldest-random	Oldest-JW	JW	Random
NONE	70	172	153	37
Caching	399	1182	1005	326
Backjumping	>10,000	>10,000	>10,000	>10,000
Semantic Branching	2087	—	—	319
BCP	90	431	616	40
Normalisation	87	207	162	39

More Optimizations for Tableau Reasoning

- Lazy unfolding—don't expand definitions too early.
- Absorption—turn disjunctive axioms into definitions.
- Modify optimizations to work for OWL—better blocking.
- Hypertableau–expansive version of tableau with more general rules.
- Global control.
- Fast one-sided subsumption tests.

Current Ontology Classification Times

	NCI-2	Plants	SWEET-P	DOLCE-P
Ontology Characteristics				
Language	ALCH	SHIF	SHOIN	SHOIN
Classes	70,576	19,145	1,728	118
Properties	189	82	145	264
Axioms	100,304	35,770	2,419	265
Sub. tests	$> 10^9$	$> 10^8$	$> 10^6$	$> 10^4$
Classification Time				
HermiT	—	11.2s	11.2s	105.1s
Pellet	172.0s	87.2s	—	105.1s
FaCT++	60.7s	22.9s	0.2s	—

Lessons Learned

- It actually works!
  - Effective reasoners for ontology reasoning in expressive languages.
  - Very impressive speedups from combinations of simple optimizations.
    - Heat death of the universe vs right now
- Why it works so well:
  - Ontologies are not *really* all that large.
  - Ontology reasoning is broken down into many small pieces, and some of the pieces are very tricky, so overcoming the tricky ones has a very large impact.
- Cleverness beats cleanliness!
  - New kinds of optimizations much more effective than faster data structures.

Mission Accomplished!?

Table of Contents

What about Reasoning with Data?

Still problematic for expressive ontologies in OWL.

- Current reasoners only reliably handle moderate amounts of data:
  - Thousands of facts or so (sometimes millions).
- Can fail for just slightly larger amounts of data.
- Why?
  - Ontology reasoning only uses a small portion of the ontology at once.
  - Data reasoning can easily use all the data at once.
- But isn't data (almost entirely) conjunctive?
  - Yes, but ontology disjunctions can intrude:
    - $Arm \equiv LeftArm \sqcup RightArm$.
  - Not really, as disjunction can be easily hidden:
    - $Alex \in (\geq 2 hasDisease) \sqcap (\leq 5 hasDisease)$

This is the current core challenge in ontology reasoning.

Current State of the Art—Evading the Problem

- Use a simpler ontology language and specialized reasoner:
  - OWL 2 QL—translate OWL queries into DB queries.
  - OWL 2 EL—guaranteed poly-time reasoning.
  - OWL 2 RL—use rule engines to expand KB and then read off answers.
  - Restricted expressivity, but complete.
  - Requires limiting the form of the ontology.
- Use all of OWL and only do incomplete reasoning:
  - Rule reasoners such as Oracle's, Sesame, Jena, ...
  - Do do something outside OWL RL, but just what?
  - Some are not even complete for OWL RL.
  - Problems with changing data, as well.
- Generally have to decide which approach to take (too) early in the design process.
  - Choice affects modelling, data, ...

Potential Solutions to the Core Challenge

1. Make OWL reasoners truely plug-and-play.
2. Build a reasoner for expressive ontologies (i.e., that use all of OWL) that can handle lots of data.

Compatible Reasoners

- A suite of reasoners with differing capabilities:
  - A reasoner for simple ontologies with very large amounts of data.
  - A reasoner for more-complex ontologies with lots of data.
  - An incomplete reasoner for complex ontologies, but fast.
    - Exploit pre-processing to reduce incompleteness.
  - A complete reasoner for complex ontologies.
    - Could only handle moderate amounts of data.
- Permit replacement very late in design cycle (even at run time).
  - Tricky issues related to modelling changes to support limited expressivity.
- Can use existing reasoners, but changes are required:
  - To support exactly the same constructs.
  - To support same interfaces.

Infrastructure for Compatible Reasoners

- Some infrastructure already exists:
  - Single (?) syntax—RDF, OWL.
  - Ontology analysis (which profile, which language constructs).
  - Common interfaces (OWL API, OWLlink).
- Need more infrastructure:
  - Reasoner selection, by characteristics of the ontology and data.
  - Ontology simplification, e.g., from OWL to OWL RL.
    - Known to be a hard problem.
  - Analysis of incompleteness:
    - When is an incomplete reasoner complete?
  - Preprocessing to reduce incompleteness:
    - E.g., do classification in original ontology and use results when data reasoning in a simplification.

Better OWL Reasoners

Advanced rule-based reasoners:

- OWL RL is designed for rule-based reasoning.
- Several systems exist—Jena, ELLY, ...
- Standard rule systems have expressivity limitations:
  - With extra individuals, e.g., every person has two parents who are people.
  - With disjunctions.
- Can we extend the rule paradigm effectively?
  - Advanced rule formalisms exist (e.g., SILK), but have not been tailored for OWL reasoning.
  - Repairing incompleteness in rule reasoners by adding in extra information derived from complete classification of ontology.

Better OWL Reasoners

Parallel reasoning:

- Gets around some problems encountered in current reasoners:
  - Provides much more processing power.
  - Provides Much more main memory.
- But current reasoners have sophisticated central control:
  - to pick best place to work next,
  - to avoid redoing work,
  - to check for loops.
- How can this be decentralized?
  - Need to check for loops even when doing deterministic reasoning.
- What is the appropriate model?
  - Not MapReduce, or even Haloop.
  - Perhaps graph programming? (Pregel toolkit from Google)

Table of Contents

Challenges in the Core of Ontology Support Systems

Build reasoners that can support ontologies and large amounts of data

- Fully-compatible reasoners
  - No reasoner works well in all situations.
  - Pick the best reasoner.
- Improved reasoners—rule based, parallel
  - Develop a reasoner that can reliably handle complex ontologies in conjunction with large amounts of data.

Other Challenges in Ontology Support Systems

A Full Ontology Support System—incorporating

- Use of compatible different reasoners handling different languages
- Ontology analysis to select reasoner
- Interacting with data
- Integrity constraints
- Hybrid reasoning, e.g., including arbitrary rules
- Programming language interfaces, e.g., with object-oriented languages
- Learning of ontologies from data
- Data source analysis and selection
- Ontology development by non-logicians
- Ontology life-cycle management

Selected Publications

Dmitry Tsarkov, Ian Horrocks, and Peter F. Patel-Schneider. Optimising Terminological Reasoning for Expressive Description Logics. *Journal of Automated Reasoning*, **39**:3, October 2007, pp. 277–316.
`http://ect.bell-labs.com/who/pfps/publications/paper-expressive.pdf`

Peter F. Patel-Schneider and Roberto Sebastiani. A New General Method to Generate Random Modal Formulae for Testing Decision Procedures. *Journal of Artificial Intelligence Research*, **18**, 2003, pages 351–389.
`http://ect.bell-labs.com/who/pfps/publications/generate-new-method.ps`

Ian Horrocks and Peter F. Patel-Schneider. Evaluating Optimised Decision Procedures for Propositional Modal Km Satisfiability. *Journal of Automated Reasoning*, **28**:2, February 2002, pages 173–204.
`http://ect.bell-labs.com/who/pfps/publications/evaluating-decision.pdf`

Ian Horrocks, Peter Patel-Schneider, and Roberto Sebastiani. An Analysis of Empirical Testing for Modal Decision Procedures. *logic Journal of the IGPL*, **8**:3, May 2000, pages 293–324.
`http://ect.bell-labs.com/who/pfps/publications/analysis-empirical.pdf`

Ian Horrocks and Peter F. Patel-Schneider. Optimising Description Logic Subsumption. *Journal of Logic and Computation*, **9**:3, June 1999, pages 267–293. `http://ect.bell-labs.com/who/pfps/publications/optimising-dl.pdf`

Ian Horrocks and Peter F. Patel-Schneider. Optimising Propositional Modal Satisfiability for Description Logic Subsumption. In *Proceedings of the Conference on Artificial Intelligence and Symbolic Computation*. Lecture Notes in Artificial Intelligence No. 1476. Springer: Berlin, September 1998.

Ian Horrocks and Peter F. Patel-Schneider. Comparing Subsumption Optimizations. In *Proceedings of the 1998 International Workshop on Description Logics*. Trento, Italy, June 1998, pages 90–94.