An Introduction to OWL

Sean Bechhofer
School of Computer Science
University of Manchester, UK
http://www.cs.manchester.ac.uk

OWL: Web Ontology Language

- OWL is an ontology language designed for the Semantic Web
  - It provides a rich collection of operators for forming concept descriptions
  - It is a W3C standard, promoting interoperation and sharing between applications
  - It has been designed to be compatible with existing web standards
- In this talk, we'll see some of the motivation behind OWL and some details of the language
The Semantic Web Vision

- The Web was made possible through established standards
  - TCP/IP for transporting bits down a wire
  - HTTP & HTML for transporting and rendering hyperlinked text
- Applications able to exploit this common infrastructure
  - Result is the WWW as we know it
- 1st generation web mostly handwritten HTML pages
- 2nd generation (current) web often machine generated/active
  - Both intended for direct human processing/interaction
- In next generation web, resources should be more accessible to automated processes
  - To be achieved via semantic markup
  - Metadata annotations that describe content/function

What’s the Problem?

- Consider a typical web page
- Markup consists of:
  - rendering information (e.g., font size and colour)
  - Hyper-links to related content
- Semantic content is accessible to humans but not (easily) to computers...
- Requires (at least) NL understanding
A Semantic Web — First Steps

- Make web resources more accessible to automated processes
- Extend existing rendering markup with semantic markup
  - Metadata annotations that describe content/function of web accessible resources
- Use Ontologies to provide vocabulary for annotations
  - New terms can be formed by combining existing ones
  - “Formal specification” is accessible to machines
- A prerequisite is a standard web ontology language
  - Need to agree common syntax before we can share semantics
  - Syntactic web based on standards such as HTTP and HTML

Technologies for the Semantic Web

- Metadata
  - Resources are marked-up with descriptions of their content. No good unless everyone speaks the same language;
- Terminologies
  - provide shared and common vocabularies of a domain, so search engines, agents, authors and users can communicate. No good unless everyone means the same thing;
- Ontologies
  - provide a shared and common understanding of a domain that can be communicated across people and applications, and will play a major role in supporting information exchange and discovery.
Building a Semantic Web

- Annotation
  - Associating metadata with resources
- Integration
  - Integrating information sources
- Inference
  - Reasoning over the information we have.
  - Could be light-weight (taxonomy)
  - Could be heavy-weight (logic-style)
- Interoperation and Sharing are key goals

Languages

- Work on Semantic Web has concentrated on the definition of a collection or “stack” of languages.
  - These languages are then used to support the representation and use of metadata.
- The languages provide basic machinery that we can use to represent the extra semantic information needed for the Semantic Web
  - XML
  - RDF
  - RDF(S)
  - OWL
  - …
Object Oriented Models

- Many languages use an “object oriented model” with
- **Objects/Instances/Individuals**
  - Elements of the domain of discourse
- **Types/Classes/Concepts**
  - Sets of objects sharing certain characteristics
- **Relations/Properties/Roles**
  - Sets of pairs (tuples) of objects
- Such languages are/can be:
  - Well understood
  - Formally specified
  - (Relatively) easy to use
  - Amenable to machine processing

Structure of an Ontology

Ontologies typically have two distinct components:
- **Names** for important concepts in the domain
  - Paper is a concept whose members are a kind of animal
  - Person is a concept whose members are persons
- **Background knowledge/constraints** on the domain
  - A Paper is a kind of ArgumentativeDocument
  - All participants in a Workshop must be Persons.
  - No individual can be both an InProceedings and a Journal
Formal Languages

- The degree of formality of ontology languages varies widely
- Increased formality makes languages more amenable to machine processing (e.g. automated reasoning).
- The formal semantics provides an unambiguous interpretation of the descriptions.

Why Semantics?

- What does an expression in an ontology mean?
- The semantics of a language can tell us precisely how to interpret a complex expression.
- Well defined semantics are vital if we are to support machine interpretability
  - They remove ambiguities in the interpretation of the descriptions.
### RDF

- RDF stands for **Resource Description Framework**
- It is a W3C Recommendation
  - [http://www.w3.org/RDF](http://www.w3.org/RDF)
- RDF is a **graphical formalism** (+ XML syntax)
  - for representing metadata
  - for describing the semantics of information in a machine-accessible way
- Provides a simple data model based on triples.

---

### The RDF Data Model

- Statements are `<subject, predicate, object>` triples:
  - `<Sean, hasColleague, Uli>`
- Can be represented as a graph:

```
  Sean  hasColleague  Uli
```

- Statements describe properties of **resources**
  - Resources are identified by URIs.
- Properties themselves are also resources (URIs)
  - Thus we can also say things about properties.
### Linking Statements

- The subject of one statement can be the object of another
- Such collections of statements form a directed, labeled graph

![Graph Diagram]

- Note that the object of a triple can also be a “literal” (a string)

---

### RDF Syntax

- RDF has a number of different concrete syntaxes
  - RDF/XML
  - N3
  - NTriples
  - Turtle
- These all give some way of serializing the RDF graph.
What does RDF give us?

- A mechanism for annotating data and resources.
- Single (simple) data model.
- Syntactic consistency between names (URIs).
- Low level integration of data.

RDF(S): RDF Schema

- RDF gives a formalism for meta data annotation, and a way to write it down, but it does not give any special meaning to vocabulary such as subClassOf or type
  - Interpretation is an arbitrary binary relation
- RDF Schema extends RDF with a schema vocabulary that allows you to define basic vocabulary terms and the relations between those terms
  - Class, type, subClassOf,
  - Property, subPropertyOf, range, domain
  - it gives "extra meaning" to particular RDF predicates and resources
  - this "extra meaning", or semantics, specifies how a term should be interpreted
RDF(S) Examples

- RDF Schema terms (just a few examples):
  - Class; Property
  - type; subClassOf
  - range; domain
- These terms are the RDF Schema building blocks (constructors) used to create vocabularies:
  - `<Person,type,Class>`
  - `<hasColleague,type,Property>`
  - `<Professor,subClassOf,Person>`
  - `<Carole,type,Professor>`
  - `<hasColleague,range,Person>`
  - `<hasColleague,domain,Person>`

RDF/RDF(S) “Liberality”

- No distinction between classes and instances (individuals)
  - `<Species,type,Class>`
  - `<Lion,type,Species>`
  - `<Leo,type,Lion>`
- Properties can themselves have properties
  - `<hasDaughter,subPropertyOf,hasChild>`
  - `<hasDaughter,type,familyProperty>`
- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other
  - `<type,range,Class>`
  - `<Property,type,Class>`
  - `<type,subPropertyOf,subClassOf>`
RDF/RDF(S) Semantics

- RDF semantics given by RDF Model Theory (MT)
  - IR, a non-empty set of resources
  - IS, a mapping from V into IR
  - IP, a distinguished subset of IR (the properties)
  - IEXT, a mapping from IP into the powerset of IR ε IR
- Class interpretation ICEXT induced by IEXT(IS(type))
  - ICEXT(C) = \{x | (x,C) \in IEXT(IS(type))\}
- RDF(S) adds constraints on models
  - \((x,y), (y,z) \models IEXT(IS(subClassOf)) \land (x,z) \notin IEXT(IS(subClassOf))\)

RDF(S) Inference
RDF(S) Inference

What does RDF(S) give us?

- Ability to use simple schema/vocabularies when describing our resources.
- Consistent vocabulary use and sharing.
- Simple inference
Problems with RDF(S)

• RDF(S) is too weak to describe resources in sufficient detail
  – No localised range and domain constraints
    • Can’t say that the range of publishedBy is Publisher when applied to Journal and Institution when applied to TechnicalReport
  – No existence/cardinality constraints
    • Can’t say that all instances of Paper have an author that is also a Person, or that Papers must have at least 3 reviewers
  – No transitive, inverse or symmetrical properties
    • Can’t say that isSubEventOf is a transitive property, or that hasRole is the inverse of isRoleAt

• Difficult to provide reasoning support
  – No “native” reasoners for non-standard semantics
  – May be possible to reason via FO axiomatisation

Solution

• Extend RDF(S) with a language that has the following desirable features identified for Web Ontology Language
  – Extends existing Web standards
    • Such as XML, RDF, RDFS
  – Easy to understand and use
    • Should be based on familiar KR idioms
  – Of “adequate” expressive power
  – Formally specified
    • Possible to provide automated reasoning support

• That language is OWL.
The OWL Family Tree

A Brief History of OWL

- **OIL**
  - Developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - Based on frame-based language
  - Strong emphasis on formal rigour.
  - Semantics in terms of Description Logics
  - RDFS based syntax
A Brief History of OWL

- DAML-ONT
  - Developed by DAML Programme.
    - Largely US based researchers
  - Extended RDFS with constructors from OO and frame-based languages
  - Rather weak semantic specification
    - Problems with machine interpretation
    - Problems with human interpretation

- DAML+OIL
  - Merging of DAML-ONT and OIL
  - Basically a DL with an RDFS-based syntax.
  - Development was carried out by “Joint EU/US Committee on Agent Markup Languages”
  - Extends ("DL subset" of) RDF

- DAML+OIL submitted to W3C as basis for standardisation
  - Web-Ontology (WebOnt) Working Group formed
A Brief History of OWL

• OWL
  – W3C Recommendation (February 2004)
  – Based largely on the DAML+OIL specification from March 2001.
  – Well defined RDF/XML serializations
  – Formal semantics
    • First Order
    • Relationship with RDF
  – Comprehensive test cases for tools/implementations
  – Growing industrial takeup.

OWL Layering

• Three species of OWL
  – OWL Full is the union of OWL syntax and RDF
  – OWL DL restricted to FOL fragment (¼ DAML+OIL)
    • Corresponds to SHOIN(DL) Description Logic
  – OWL Lite is "simpler" subset of OWL DL

• Syntactic Layering

• Semantic Layering
  – OWL DL semantics = OWL Full semantics (within DL fragment)
  – OWL Lite semantics = OWL DL semantics (within Lite fragment)

• DL semantics are definitive
  – In principle: correspondence proof
  – But: if Full disagrees with DL (in DL fragment), then Full is wrong
OWL Full

- No restriction on use of OWL vocabulary (as long as legal RDF)
  - Classes as instances (and much more)
- RDF style model theory
  - Reasoning using FOL engines
    - via axiomatisation
  - Semantics should correspond with OWL DL for suitably restricted KBs

OWL DL

- Use of OWL vocabulary restricted
  - Can’t be used to do “nasty things” (i.e., modify OWL)
  - No classes as instances
  - Defined by abstract syntax + mapping to RDF
- Standard DL/FOL model theory (definitive)
  - Direct correspondence with (first order) logic
- Benefits from years of DL research
  - Well defined semantics
  - Formal properties well understood (complexity, decidability)
  - Known reasoning algorithms
  - Implemented systems (highly optimised)
**OWL Lite**

- Like DL, but fewer constructs
  - No explicit negation or union
  - Restricted cardinality (zero or one)
  - No nominals (oneOf)
- Semantics as per DL
  - Reasoning via standard DL engines (+datatypes)
    - E.g., FaCT, RACER, Cerebra, Pellet
- In practice, not really used.
  - Possible alternative: "tractable fragments"

---

**Aside: Description Logics**

- A family of logic based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), roles (relationships) and individuals
- Distinguished by:
  - **Formal semantics** (typically model theoretic)
    - Decidable fragments of FOL
    - Closely related to Propositional Modal & Dynamic Logics
  - ** Provision of inference services**
    - Sound and complete decision procedures for key problems
    - Implemented systems (highly optimised)
DL Semantics

• Model theoretic semantics. An interpretation consists of
  – A domain of discourse (a collection of objects)
  – Functions mapping
    • classes to sets of objects
    • properties to sets of pairs of objects
  – Rules describe how to interpret the constructors and tell us when
    an interpretation is a model.
• In a DL, a class description is thus a characterisation of the
  individuals that are members of that class.

OWL Syntaxes

• Abstract Syntax
  – Used in the definition of the language and the DL/Lite semantics
• OWL in RDF (the “official” concrete syntax)
  – RDF/XML presentation
• XML Presentation Syntax
  – XML Schema definition
OWL Class Constructors

• OWL has a number of operators for constructing class expressions.
• These have an associated semantics which is given in terms of a domain:
  – \( \Delta \)
• And an interpretation function
  – \( I \):concepts ! \( \wp(\Delta) \)
  – \( I \):properties ! \( \wp(\Delta \times \Delta) \)
  – \( I \):individuals ! \( \Delta \)
• \( I \) is then extended to concept expressions.

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Example</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>Human</td>
<td>( I(\text{Human}) )</td>
</tr>
<tr>
<td>intersectionOf</td>
<td>intersectionOf(Human Male)</td>
<td>( I(\text{Human}) \land I(\text{Male}) )</td>
</tr>
<tr>
<td>unionOf</td>
<td>unionOf(Doctor Lawyer)</td>
<td>( I(\text{Doctor}) \setminus I(\text{Lawyer}) )</td>
</tr>
<tr>
<td>complementOf</td>
<td>complementOf(Male)</td>
<td>( \Delta \setminus I(\text{Male}) )</td>
</tr>
<tr>
<td>oneOf</td>
<td>oneOf(john mary)</td>
<td>{I(john), I(mary)}</td>
</tr>
</tbody>
</table>
**OWL Class Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Example</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>someValuesFrom</td>
<td>restriction(hasChild someValuesFrom Lawyer)</td>
<td>${x \exists y. x, y \models L(hasChild) \land y \models L(Lawyer)}$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>restriction(hasChild allValuesFrom Doctor)</td>
<td>${x \forall y. x, y \models L(hasChild) \land y \models L(Doctor)}$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>restriction(hasChild minCardinality (2))</td>
<td>${x \exists #x, y \models L(hasChild) \land y \models L(Doctor)}$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>restriction(hasChild maxCardinality (2))</td>
<td>${x \forall #x, y \models L(hasChild) \land y \models L(Doctor)}$</td>
</tr>
</tbody>
</table>

**OWL Axioms**

- Axioms allow us to add further statements about arbitrary concept expressions and properties
  - Subclasses, Disjointness, Equivalence, transitivity of properties etc.
- An interpretation is then a model of the axioms iff it satisfies every axiom in the model.

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Example</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubClassOf</td>
<td>SubClassOf(Human Animal)</td>
<td>$I(Human) \subseteq I(Animal)$</td>
</tr>
<tr>
<td>EquivalentClasses</td>
<td>EquivalentClass(Man intersectionOf(Human Male))</td>
<td>$I(Man) = I(Human) \cap I(Male)$</td>
</tr>
<tr>
<td>DisjointClasses</td>
<td>DisjointClasses(Animal Plant)</td>
<td>$I(Animal) \cap I(Plant) = \emptyset$</td>
</tr>
</tbody>
</table>
# OWL Individual Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Example</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual</td>
<td>Individual(Sean type(Human))</td>
<td>I(Sean) : I(Human)</td>
</tr>
<tr>
<td>Individual</td>
<td>Individual(Sean value(worksWith Uli))</td>
<td>⊨ I(Sean), I(Uli) ⇒ I(worksWith)</td>
</tr>
<tr>
<td>DifferentIndividuals</td>
<td>DifferentIndividuals(Sean Uli)</td>
<td>I(Sean) ≠ I(Uli)</td>
</tr>
<tr>
<td>SameIndividualAs</td>
<td>SameIndividualAs(George.W.Bush PresidentBush)</td>
<td>I(George.W.Bush) = I(PresidentBush)</td>
</tr>
</tbody>
</table>

# OWL Property Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Example</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubPropertyOf</td>
<td>SubPropertyOf(hasMother hasParent)</td>
<td>I(hasMother) ⇒ I(hasParent)</td>
</tr>
<tr>
<td>domain</td>
<td>ObjectProperty (owns domain(Person))</td>
<td>$\forall x,\forall y,\forall z (\text{owns}) \Rightarrow \forall z (\text{Person})</td>
</tr>
<tr>
<td>range</td>
<td>ObjectProperty (employs range(Person))</td>
<td>$\forall x,\forall y,\forall z (\text{employs}) \Rightarrow \forall y (\text{Person})</td>
</tr>
</tbody>
</table>
| transitive             | ObjectProperty(hasPart Transitive)              | $\forall x,\forall y,\forall z (\text{hasPart}) \Rightarrow \forall z (\text{hasPart})$
Semantics

- An interpretation $I$ satisfies an axiom if the interpretation of the axiom is true.
- $I$ satisfies or is a model of an ontology (or knowledge base) if the interpretation satisfies all the axioms in the knowledge base (class axioms, property axioms and individual axioms).
- $C$ subsumes $D$ w.r.t. an ontology $O$ iff for every model $I$ of $O$, $I(D) \subseteq I(C)$
- $C$ is equivalent to $D$ w.r.t. an ontology $O$ iff for every model $I$ of $O$, $I(C) = I(D)$
- $C$ is satisfiable w.r.t. $O$ iff there exists some model $I$ of $O$ s.t. $I(C) \neq 0$
- An ontology $O$ is consistent iff there exists some model $I$ of $O$.

Reasoning

- A reasoner makes use of the information asserted in the ontology.
- Based on the semantics described, a reasoner can help us to discover inferences that are a consequence of the knowledge that we’ve presented that we weren’t aware of beforehand.
- Is this new knowledge?
  - What’s actually in the ontology?
Reasoning

- **Subsumption reasoning**
  - Allows us to infer when one class is a subclass of another
  - B is a subclass of A if it is necessarily the case that (in all models), all instances of B must be instances of A.
  - This can be either due to an explicit assertion, or through some inference process based on an intensional definition.
  - Can then build concept hierarchies representing the taxonomy.
  - This is classification of classes.

- **Satisfiability reasoning**
  - Tells us when a concept is unsatisfiable
    - i.e. when there is no model in which the interpretation of the class is non-empty.
  - Allows us to check whether our model is consistent.

---

Necessary and Sufficient Conditions

- Classes can be described in terms of necessary and sufficient conditions.
  - This differs from some frame-based languages where we only have necessary conditions.

- **Necessary conditions**
  - Must hold if an object is to be an instance of the class

- **Sufficient conditions**
  - Those properties an object must have in order to be recognised as a member of the class.
  - Allows us to perform automated classification.
Example

Class: Paper
SubClassOf:
  author min 1

- All Papers must have at least one author
- This is a necessary condition on being a Paper, but doesn’t give us sufficiency conditions.

Introduction to OWL

Example

Class: GoodPaper
EquivalentTo:
  Paper
  and author some (Person
  and member some KoreanInstitute)

- A GoodPaper is one with an author from a KoreanInstitute
- This provides necessary and sufficient conditions for being a GoodPaper. If we know it is a Paper and there is an author from a KoreanInstitute, then it is a GoodPaper
Reasoning

- We can now infer that Paper1 is a GoodPaper

Example

- A VeryGoodPaper is one with only authors from a KoreanInstitute
- This again provides necessary and sufficient conditions for being a VeryGoodPaper. If we know it is a Paper and that all the authors are from a KoreanInstitute, then it is a VeryGoodPaper
- We can also now infer that all VeryGoodPapers are GoodPapers
Closed and Open Worlds

• The standard semantics of OWL makes an Open World Assumption (OWA).
  – We cannot assume that all information is known about all the individuals in a domain.
  – Facilitates reasoning about the intensional definitions of classes.
  – Sometimes strange side effects

• Closed World Assumption (CWA)
  – Named individuals are the only individuals in the domain

• Negation as failure.
  – If we can’t deduce that x is an A, then we know it must be a (not A).
  – Facilitate reasoning about a particular state of affairs.

Open Worlds

• Is this a VeryGoodPaper?
• We don’t know!
• Just because it is not stated that BobDylan is a member of a KoreanInstitute, we cannot assume that this is not the case.
• Similarly, there may be other authors of the paper that we do not know about.
Open Worlds

Individual: Paper3
Types: Paper
Facts:
  author KimHyunJung
  author NeilYoung

Individual: DancePopUniversity
Types: KoreanInstitute

Individual: NeilYoung
Types: Person
member max 1
Facts:
  member: UniversityOfRock

Individual: UniversityOfRock
Types:
  not KoreanInstitute

• Is this a VeryGoodPaper?
  • No!
  • Here we know for sure that NeilYoung isn’t a member of a KoreanInstitute.

Open Worlds

Individual: Paper4
Types: Paper
author max 2
Facts:
  author KimHyunJung
  author SunHoYoung

Individual: DancePopUniversity
Types: KoreanInstitute

Individual: SunHoYoung
Types: Person
Facts:
  member: KPopInstitute

Individual: KPopInstitute
Types: KoreanInstitute

• Is this a VeryGoodPaper?
  • Yes!
  • We know that all authors are from KoreanInstitutes
Why Reasoning?

- Reasoning can be used as a design support tool
  - Check **logical consistency** of classes
  - Compute implicit class hierarchy
- May be less important in small local ontologies
  - Can still be useful tool for design and maintenance
  - Much more important with larger ontologies/multiple authors
- Valuable tool for integrating and sharing ontologies
  - Use definitions/axioms to establish inter-ontology relationships
  - Check for **consistency** and (unexpected) implied relationships
- For most DLs, the basic inference problems are **decidable** (e.g. there is some program that solves the problem in a finite number of steps)

Extensions

- OWL is not intended to be the answer to all our problems.
- There are things that we can’t represent using OWL.
- Current work on extending OWL includes:
  - Rules
    - RIF
  - Extending expressivity (within certain bounds)
    - OWL1.1
  - Query
    - SPARQL
Extensions: Rules

- W3C Group chartered with producing a Rules Interchange Format
  [http://www.w3.org/2005/rules/](http://www.w3.org/2005/rules/)
- Current status
  - Use cases and Requirements
  - RIF Core Design
  - Large and somewhat disparate group
  - Production Rules, Business Rules, First Order Logic.....

Extensions: OWL1.1

- A number of domains require expressivity that is not in the current OWL specification
  - Driven by User Requirements and technical advances
  - OWLEd series of workshops
- Much of this functionality can be added in a principled way that preserves the desirable properties of OWL (DL).
- OWL Working Group now chartered:
Extensions: OWL 1.1

- **Syntactic Sugar**
  - DisjointUnion
  - Negated Property assertions
- **Richer Datatypes**
- **Complex Role Axioms**
  - Role inclusion
- **Metamodelling and Annotations**
  - Punning
- **Tractable Fragments**
  - Language fragments with desirable computational complexity

OWL1.1: Role Axioms

- Many applications (for example medicine) have requirements to specify interactions between roles:
  - A fracture located in part of the Femur is a fracture of the Femur.
- We **cannot** express such general patterns in OWL.
- Algorithms have been developed to support sound and complete reasoning in a DL extended with complex role inclusions
OWL1.1: Metamodelling

- OWL DL has strict rules about separation of namespaces.
- A URI cannot be typed as both a class and individual in the same ontology.
- OWL 1.1 allows punning, where a URI can be used in multiple roles.
  - However, the use of the URI as an individual has no bearing on the use of the URI as a class.
  - Requires explicit context telling us the role that a URI is playing.

OWL1.1: Fragments

- EL++
  - Medical Ontologies
  - SNOMED/GALEN
- DL Lite
  - Tailored for handling large numbers of facts
  - Efficient Querying
- DLP
  - Subset of OWL DL and Horn Logic
  - OWL semantics
- Horn-SHIQ
  - Similar to DLP
- RDF Schema
  - RDFS ontologies that are valid OWL1.1
Extensions: Query and Retrieval

- In standard DLs, reasoning is split into:
  - T-Box: reasoning about classes
  - A-Box: reasoning about instances
- T-Box reasoning is well understood, at least for languages like SHIQ (~OWL Lite)
  - E.g. subsumption & satisfiability testing
- Full A-Box reasoning is much more challenging
  - E.g. instance retrieval & instantiation
Query Languages

- SPARQL is a proposed query language for RDF.
  - [http://www.w3.org/TR/rdf-sparql-query/](http://www.w3.org/TR/rdf-sparql-query/)
- SPARQL Protocol, Query Language and results format.
- Query language is the interesting bit
  - Protocol allows query, no update
  - Variety of results formats: XML, JSON (used in web 2.0 apps), and RDF

SPARQL

- QL is a Candidate Recommendation as of June 14th
- Implementations
  - Jena
  - Sesame
  - Virtuoso
  - Boca
  - …
- Tightening of the spec since last year
  - In particular, the adoption of a clear algebra
SPARQL for OWL

- SPARQL for OWL
- OWL’s standard syntax is RDF
- Several implementations use SPARQL for conjunctive ABox query
  - E.g., Pellet, KAON2
- Many issues
  - Inference related, e.g., dealing with contradictions
  - Expectations
    - SPARQL users expect to query schema as well as data
    - Traditional DL query separates them

Tools

- Editors
  - Protégé OWL, SWOOP, ICOM, TopQuadrant Composer, OntoTrack, POWL, NeOn...
  - Tend to present the user with “frame-like” interfaces, but allow richer expressions
- Reasoners
  - DL style reasoners based on tableaux algorithms
    - Racer, FaCT++, Pellet
  - Based on rules or F-logic
    - F-OWL, E-Wallet.....
- APIs and Frameworks
  - Jena, WonderWeb OWL-API, KAON2, Protégé OWL API, OWLIM
Summary

• OWL provides us with a rich language for defining ontologies.
  – Builds upon RDF and RDF Schema
  – Formal semantics
    • Provides an unambiguous interpretation of expressions and facilitates the use of reasoners.
    • Draws on years of DL research.
  – Language extensions in the pipeline.
• A growing body of experience and take up in applications

Acknowledgements

• Many thanks to all the people who I "borrowed" material from, in particular
  – Ian Horrocks, Frank van Harmelen, Alan Rector, Nick Drummond, Matthew Horridge, Uli Sattler, Bijan Parsia
• and thanks to all those that they borrowed material from!
  – Too many to mention…