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
Towards a Semantic Web

- 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 defined 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
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:

• 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

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.
- Linked Data (to come….)
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 us 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)
  
  \[
  \langle \text{Species}, \text{type}, \text{Class} \rangle \\
  \langle \text{Lion}, \text{type}, \text{Species} \rangle \\
  \langle \text{Leo}, \text{type}, \text{Lion} \rangle
  \]

- Properties can themselves have properties
  
  \[
  \langle \text{hasDaughter}, \text{subPropertyOf}, \text{hasChild} \rangle \\
  \langle \text{hasDaughter}, \text{type}, \text{familyProperty} \rangle
  \]

- No distinction between language constructors and ontology vocabulary, so constructors can be applied to themselves/each other
  
  \[
  \langle \text{type}, \text{range}, \text{Class} \rangle \\
  \langle \text{Property}, \text{type}, \text{Class} \rangle \\
  \langle \text{type}, \text{subPropertyOf}, \text{subClassOf} \rangle
  \]
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 x IR

- Class interpretation ICEXT induced by IEXT(IS(type))
  - ICEXT(C) = {x | (x,C) ∈ IEXT(IS(type))}

- RDF(S) adds constraints on models
  - {(x,y), (y,z)} ⊆ IEXT(IS(subClassOf)) ⇒ (x,z) ∈ IEXT(IS(subClassOf))
RDF(S) Inference

```
Person rdfs:subClassOf Academic
Person rdf:type rdfs:Class
Person rdf:type Person
Person rdf:type Academic
Person rdf:type Lecturer

Academic rdfs:subClassOf Lecturer
Academic rdf:type rdfs:Class
Academic rdf:type Academic
Academic rdf:type Lecturer

Lecturer rdf:type rdfs:Class
Lecturer rdf:type Lecturer
Lecturer rdf:type Person
```
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`
- Can be difficult to provide **reasoning support**
  - 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

- RDF/RDF(S)
- DAML
- DAML-ONT
- OIL
- DAML+OIL
- OWL
- Frames
- Description Logics

Joint EU/US Committee

OntoKnowledge+Others

W3C
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.
There are three “species” of OWL:
- OWL Full
- OWL DL
- OWL Lite

Syntactic Layering

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

- No restriction on use of OWL vocabulary (as long as legal RDF)
  - Classes as instances (and much more)
- RDF style model theory
  - 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.
OWL Syntax

- 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
- Various “Human Readable” Syntaxes
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
  - $I:\text{concepts} \rightarrow \wp(\Delta)$
  - $I:\text{properties} \rightarrow \wp(\Delta \times \Delta)$
  - $I:\text{individuals} \rightarrow \Delta$
- $I$ is then extended to concept expressions.
# OWL Class Constructors

<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>$I(\text{Human}) \cap I(\text{Male})$</td>
</tr>
<tr>
<td>unionOf</td>
<td>unionOf(Doctor Lawyer)</td>
<td>$I(\text{Doctor}) \cup I(\text{Lawyer})$</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</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>restriction(hasChild allValuesFrom Doctor)</td>
<td>{x</td>
</tr>
<tr>
<td>minCardinality</td>
<td>restriction(hasChild minCardinality (2))</td>
<td>{x</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>restriction(hasChild maxCardinality (2))</td>
<td>{x</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>$I(\text{Human}) \subseteq I(\text{Animal})$</td>
</tr>
<tr>
<td>EquivalentClasses</td>
<td>EquivalentClass(Man intersectionOf(Human Male))</td>
<td>$I(\text{Man}) = I(\text{Human}) \cap I(\text{Male})$</td>
</tr>
<tr>
<td>DisjointClasses</td>
<td>DisjointClasses(Animal Plant)</td>
<td>$I(\text{Animal}) \cap I(\text{Plant}) = \emptyset$</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>$I(Sean) \in I(Human)$</td>
</tr>
<tr>
<td>Individual</td>
<td>Individual(Sean value(worksWith Uli))</td>
<td>$\langle I(Sean),I(Uli) \rangle \in I(worksWith)$</td>
</tr>
<tr>
<td>DifferentIndividuals</td>
<td>DifferentIndividuals(Sean Uli)</td>
<td>$I(Sean) \not= I(Uli)$</td>
</tr>
<tr>
<td>SameIndividualAs</td>
<td>SameIndividualAs(George WBush PresidentBush)</td>
<td>$I(GeorgeWBush) = I(PresidentBush)$</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>$I(\text{hasMother}) \subseteq I(\text{hasParent})$</td>
</tr>
<tr>
<td>domain</td>
<td>ObjectProperty (owns domain(Person))</td>
<td>$\forall x. \langle x,y \rangle \in I(\text{owns}) \Rightarrow x \in I(\text{Person})$</td>
</tr>
<tr>
<td>range</td>
<td>ObjectProperty (employs range(Person))</td>
<td>$\forall x. \langle x,y \rangle \in I(\text{employs}) \Rightarrow y \in I(\text{Person})$</td>
</tr>
<tr>
<td>transitive</td>
<td>ObjectProperty(hasPart Transitive)</td>
<td>$\forall x,y,z. (\langle x,y \rangle \in I(\text{hasPart}) \land \langle y,z \rangle \in I(\text{hasPart})) \Rightarrow \langle x,z \rangle \in I(\text{hasPart})$</td>
</tr>
</tbody>
</table>
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).

• The axioms in an ontology constrain the possible interpretations
• Given an ontology $O$, the constraints on the possible interpretations may lead to consequences in those interpretations.
  
  • $C$ 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. $I(C) \neq \emptyset$

  • 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.
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)
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.

If it looks like a duck and walks like a duck, then it’s a duck!
Common Misconceptions

- Disjointness of primitives
- Interpreting domain and range
- And and Or
- Quantification
- Closed and Open Worlds
Disjointness

• By default, primitive classes are not disjoint.
• Unless we explicitly say so, the description (Animal and Vegetable) is not inconsistent.
• Similarly with individuals -- the so-called Unique Name Assumption (often present in DL languages) does not hold, and individuals are not considered to be distinct unless explicitly asserted to be so.
Domain and Range

• OWL allows us to specify the **domain** and **range** of properties.
• Note that this is not interpreted as a constraint.
• Rather, the domain and range assertions allow us to make **inferences** about individuals.
• Consider the following:
  • **ObjectProperty**: employs
    Domain: Company
    Range: Person
    **Individual**: IBM
    **Facts**: employs Jim
• If we haven’t said anything else about **IBM** or **Jim**, this is **not** an error. However, we can now **infer** that **IBM** is a **Company** and **Jim** is a **Person**.
And/Or and Quantification

• The logical connectives And and Or often cause confusion
  – Tea or Coffee?
  – Milk and Sugar?

• Quantification can also be contrary to our intuition.
  – Universal quantification over an empty set is true.
  – Sean is a member of hasChild only Martian
  – Existential quantification may imply the existence of an individual that we don’t know the name of.
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 \( \text{not } A \).
  - Facilitate reasoning about a particular state of affairs.
What does OWL give us?

- A KR language that allows us to define ontologies including definitions and constraints that may involve complex expressions.
- A KR language that lives on the web.
- A well defined semantics facilitating the use of reasoning techniques.
OWL isn’t everything

- OWL is not intended to be the answer to all our problems.
- For some applications, less formal vocabularies may be more appropriate.
- For some applications, more expressiveness may be needed.
Lightweight Vocabularies

• For many applications, lightweight representations are more appropriate.
• Thesauri, classification schemes, taxonomies and other controlled vocabularies
  – Many of these already exist and are in use in cultural heritage, library sciences, medicine etc.
  – Often have some taxonomic structure, but with a less precise semantics.
SKOS: Simple Knowledge Organisation System

- SKOS aims to provide an RDF vocabulary for the representation of such schemes.
- W3C Semantic Web Deployment Group currently working towards a Recommendation for SKOS.
- Focus on Retrieval Scenarios
  A. Single controlled vocabulary used to index and then retrieve objects
  B. Different controlled vocabularies used to index and retrieve objects
     - Mappings then required between the vocabularies
     - Initial use cases/requirements focus on these tasks
       - Not worrying about activities like Natural Language translation
Concept Schemes

• A concept scheme is a set of concepts, potentially including statements about relationships between those concepts
  – Broader Terms
  – Narrower Terms
  – Related Terms
  – Synonyms, usage information etc.

• Concept schemes aren’t formal ontologies in the way that OWL ontologies are formal ontologies.
  – Relationships such as broader/narrower are not necessarily interpreted as set inclusion.
Lexical Labels

• SKOS provides a number of properties allowing labelling of concepts.
  – Preferred Labels
  – Alternative Labels (synonyms)
  – Hidden Labels (e.g. spelling mistakes useful as lead in vocabulary)

• SKOS labelling properties may also be useful in annotating OWL ontologies.
SKOS Example

prefix skos: <http://www.w3.org/2004/02/skos/core#>

'Economic policy'

'Economic cooperation'

'Interdependence'

'Economic co-operation'

Includes cooperative measures in banking, trade, industry etc., between and among countries.

'Economic integration'

'European economic cooperation'

'European industrial cooperation'

Industrial cooperation
SKOS

• Semantic Web Deployment Working Group
  http://www.w3.org/2006/07/SWD/

• SKOS Reference:
  http://www.w3.org/TR/skos-reference/

• SKOS Primer
  http://www.w3.org/TR/skos-primer/

• Documents currently in Last Call
OWL 2

• 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:

  http://www.w3.org/2007/OWL/
OWL 2

• Additional Expressivity (SROIQ)
  – Qualified Cardinality Restrictions
  – Local reflexivity restrictions
  – Reflexive/Irreflexive/Symmetric/Asymmetric properties
  – Property chains
  – Disjoint Properties
• Richer Datatypes
  – User defined datatypes
• Metamodelling and Annotations
  – Punning
• Profiles
  – Language fragments with desirable computational complexity
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.
• 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 2 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.
OWL 2 Profiles

- **OWL 2 EL**
  - Polynomial time reasoning
  - Medical Ontologies
  - SNOMED

- **OWL 2 QL**
  - Conjunctive query using conventional relation db systems
  - Tailored for handling large numbers of facts
  - Efficient Querying

- **OWL 2 RL**
  - Forward chaining rules.
Tools

• Editors
  – Protégé OWL, SWOOP, ICOM, TopQuadrant Composer, OntoTrack, NeOn. Altova SemanticWorks…
  – 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.
- A KR Language for the Web
- Language extensions under development
- 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…