Tutorial on OWL

ISWC, Sanibel Island, Florida, USA

20th October, 2003

Sean Bechhofer,\textsuperscript{1} Ian Horrocks\textsuperscript{1} and Peter F. Patel-Schneider\textsuperscript{2}

\textsuperscript{1}University of Manchester
Manchester, UK
\{horrocks\textbackslash seanb\}@cs.man.ac.uk

\textsuperscript{2}Bell Labs Research
Murray Hill, NJ, USA
\{horrocks\textbackslash seanb\}@cs.man.ac.uk

Tutorial on OWL

Contents

• Introduction to the Semantic Web
• Example OWL Ontology
• Reasoning Services
• OilEd
Introduction to the Semantic Web

History of the Semantic Web

- Web was “invented” by Tim Berners-Lee (amongst others), a physicist working at CERN
- TBL’s original vision of the Web was much more ambitious than the reality of the existing (syntactic) Web:
  
  “... a goal of the Web was that, if the interaction between person and hypertext could be so intuitive that the machine-readable information space gave an accurate representation of the state of people’s thoughts, interactions, and work patterns, then machine analysis could become a very powerful management tool, seeing patterns in our work and facilitating our working together through the typical problems which beset the management of large organizations.”

- TBL (and others) have since been working towards realising this vision, which has become known as the Semantic Web
  - E.g., article in May 2001 issue of Scientific American...
Realising the complete “vision” is too hard for now (probably)
But we can make a start by adding semantic annotation to web resources

Where we are Today: the Syntactic Web

[Scientific American, May 2001: Hendler & Miller 02]
The Syntactic Web is…

- A hypermedia, a digital library
  - A library of documents called (web pages) interconnected by a hypermedia of links
- A database, an application platform
  - A common portal to applications accessible through web pages, and presenting their results as web pages
- A platform for multimedia
  - BBC Radio 4 anywhere in the world! Terminator 3 trailers!
- A naming scheme
  - Unique identity for those documents

A place where computers do the presentation (easy) and people do the linking and interpreting (hard).

Why not get computers to do more of the hard work?

[Gooble 03]

Hard Work using the Syntactic Web…

Find images of Peter Patel-Schneider, Frank van Harmelen and Alan Rector…

Impossible (?) using the Syntactic Web…

- Complex queries involving **background knowledge**
  - Find information about “animals that use sonar but are not either bats or dolphins”; e.g., Barn Owl

- Locating information in repositories
  - Travel enquiries
  - Prices of goods
  - Results of human genome experiments

- Finding and using “services”
  - Visualise surface interactions between two proteins

- Delegating complex tasks to web “agents”
  - Book me a holiday next weekend somewhere warm, not too far away, and where they speak French or English

What is 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…
What information can we see...

WWW2002
The eleventh international world wide web conference
Sheraton waikiki hotel
Honolulu, hawaii, USA
7-11 may 2002
1 location 5 days learn interact
Registered participants coming from
australia, canada, chile denmark, france, germany, ghana, hong kong, india,
ireland, italy, japan, malta, new zealand, the netherlands, norway,
singapore, switzerland, the united kingdom, the united states, vietnam,
zaire
Register now
On the 7th May Honolulu will provide the backdrop of the eleventh
international world wide web conference. This prestigious event ...
Speakers confirmed
Tim berners-lee
Tim is the well known inventor of the Web, ...
Ian Foster
Ian is the pioneer of the Grid, the next generation internet ...

What information can a machine see...


Solution: XML markup with “meaningful” tags?

But What About…
Need to Add “Semantics”

- **External agreement** on meaning of annotations
  - E.g., Dublin Core
    - Agree on the meaning of a set of annotation tags
  - Problems with this approach
    - Inflexible
    - Limited number of things can be expressed
- **Use Ontologies** to specify meaning of annotations
  - Ontologies provide a vocabulary of terms
  - New terms can be formed by combining existing ones
  - Meaning (semantics) of such terms is formally specified
  - Can also specify relationships between terms in multiple ontologies
Ontology: Origins and History

Ontology in Philosophy

a philosophical discipline—a branch of philosophy that deals with the nature and the organisation of reality

- Science of Being (Aristotle, Metaphysics, IV, 1)
- Tries to answer the questions:
  
  What characterizes being?
  
  Eventually, what is being?

Ontology in Linguistics

Concept

activates

Form

Relates to

Referent

"Tank"

[Ogden, Richards, 1923]
Ontology in Computer Science

- An ontology is an engineering artifact:
  - It is constituted by a specific vocabulary used to describe a certain reality, plus
  - a set of explicit assumptions regarding the intended meaning of the vocabulary.

- Thus, an ontology describes a formal specification of a certain domain:
  - Shared understanding of a domain of interest
  - Formal and machine manipulable model of a domain of interest

“An explicit specification of a conceptualisation” [Gruber93]

Structure of an Ontology

Ontologies typically have two distinct components:

- Names for important concepts in the domain
  - Elephant is a concept whose members are a kind of animal
  - Herbivore is a concept whose members are exactly those animals who eat only plants or parts of plants
  - Adult_Elephant is a concept whose members are exactly those elephants whose age is greater than 20 years

- Background knowledge/constraints on the domain
  - Adult_Elephants weigh at least 2,000 kg
  - All Elephants are either African_Elephants or Indian_Elephants
  - No individual can be both a Herbivore and a Carnivore
Example Ontology

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
  - “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
Ontology Design and Deployment

• Given key role of ontologies in the Semantic Web, it will be essential to provide tools and services to help users:
  – Design and maintain high quality ontologies, e.g.:
    • Meaningful — all named classes can have instances
    • Correct — captured intuitions of domain experts
    • Minimally redundant — no unintended synonyms
    • Richly axiomatised — (sufficiently) detailed descriptions
  – Store (large numbers) of instances of ontology classes, e.g.:
    • Annotations from web pages
  – Answer queries over ontology classes and instances, e.g.:
    • Find more general/specific classes
    • Retrieve annotations/pages matching a given description
  – Integrate and align multiple ontologies

Ontology Languages for the Semantic Web
Ontology Languages

• Wide variety of languages for “Explicit Specification”
  – Graphical notations
    • Semantic networks
    • Topic Maps (see http://www.topicmaps.org/)
    • UML
    • RDF
  – Logic based
    • Description Logics (e.g., OIL, DAML+OIL, OWL)
    • Rules (e.g., RuleML, LP/Prolog)
    • First Order Logic (e.g., KIF)
    • Conceptual graphs
    • (Syntactically) higher order logics (e.g., LBase)
    • Non-classical logics (e.g., Flogic, Non-Mon, modalities)
  – Probabilistic/fuzzy
• Degree of formality varies widely
  – Increased formality makes languages more amenable to machine processing (e.g., automated reasoning)

Many languages use “object oriented” model based on:

• Objects/Instances/Individuals
  – Elements of the domain of discourse
  – Equivalent to constants in FOL
• Types/Classes/Concepts
  – Sets of objects sharing certain characteristics
  – Equivalent to unary predicates in FOL
• Relations/Properties/Roles
  – Sets of pairs (tuples) of objects
  – Equivalent to binary predicates in FOL
• Such languages are/can be:
  – Well understood
  – Formally specified
  – (Relatively) easy to use
  – Amenable to machine processing
Web “Schema” Languages

• Existing Web languages extended to facilitate content description
  – XML → XML Schema (XMLS)
  – RDF → RDF Schema (RDFS)
• XMLS not an ontology language
  – Changes format of DTDs (document schemas) to be XML
  – Adds an extensible type hierarchy
    • Integers, Strings, etc.
    • Can define sub-types, e.g., positive integers
• RDFS is recognisable as an ontology language
  – Classes and properties
  – Sub/super-classes (and properties)
  – Range and domain (of properties)

RDF and RDFS

• RDF stands for Resource Description Framework
• It is a W3C candidate recommendation (http://www.w3.org/RDF)
• RDF is graphical formalism (+ XML syntax + semantics)
  – for representing metadata
  – for describing the semantics of information in a machine-accessible way
• RDFS extends RDF with “schema vocabulary”, e.g.:
  – Class, Property
  – type, subClassOf, subPropertyOf
  – range, domain
The RDF Data Model

- Statements are <subject, predicate, object> triples:
  - \(<Ian, hasColleague, Uli>\)
- Can be represented as a graph:

```
  Ian --- hasColleague --- Uli
```
- Statements describe properties of resources
- A resource is any object that can be pointed to by a URI:
  - a document, a picture, a paragraph on the Web;
  - a book in the library, a real person (?)
  - `isbn://5031-4444-3333`
  - ...
- Properties themselves are also resources (URIs)

URIs

- URI = Uniform Resource Identifier
- "The generic set of all names/addresses that are short strings that refer to resources"
- URLs (Uniform Resource Locators) are a particular type of URI, used for resources that can be accessed on the WWW (e.g., web pages)
- In RDF, URIs typically look like “normal” URLs, often with fragment identifiers to point at specific parts of a document:
  - `http://www.somedomain.com/some/path/to/file#fragmentID`
Linking Statements

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

![Diagram of directed labeled graph with nodes Ian, Uli, Carole and edges hasColleague and hasHomePage]

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

RDF Syntax

- RDF has an XML syntax that has a specific meaning:
- Every Description element describes a resource
- Every attribute or nested element inside a Description is a property of that Resource
- We can refer to resources by using URIs

```xml
<Description about="some.uri/person/ian_horrocks">
  <hasColleague resource="some.uri/person/uli_sattler"/>
</Description>
<Description about="some.uri/person/uli_sattler">
  <hasHomePage>http://www.cs.mam.ac.uk/~sattler</hasHomePage>
</Description>
<Description about="some.uri/person/carole_goble">
  <hasColleague resource="some.uri/person/uli_sattler"/>
</Description>
```
RDF Schema (RDFS)

• RDF gives a formalism for meta data annotation, and a way to write it down in XML, but it does not give any special meaning to vocabulary such as subClassOf or type
  – Interpretation is an arbitrary binary relation

• RDF Schema allows you to define vocabulary terms and the relations between those terms
  – it gives “extra meaning” to particular RDF predicates and resources
  – this “extra meaning”, or semantics, specifies how a term should be interpreted

RDFS 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/RDFS “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/RDFS Semantics

- RDF has “Non-standard” semantics in order to deal with this
- Semantics given by RDF Model Theory (MT)
Semantics and Model Theories

- Ontology/KR languages aim to model (part of) world
- Terms in language correspond to entities in world
- Meaning given by, e.g.:
  - Mapping to another formalism, such as FOL, with own well defined semantics
  - or a bespoke Model Theory (MT)
- **MT defines relationship between syntax and interpretations**
  - Can be many interpretations (models) of one piece of syntax
  - Models supposed to be analogue of (part of) world
    - E.g., elements of model correspond to objects in world
  - Formal relationship between syntax and models
    - Structure of models reflect relationships specified in syntax
  - Inference (e.g., subsumption) defined in terms of MT
    - E.g., $\exists I \vDash A \sqsubseteq B$ iff in every model of $I$, $\text{ext}(A) \subseteq \text{ext}(B)$

RDF/RDFS Semantics

- RDF has “Non-standard” semantics in order to deal with this
- Semantics given by RDF Model Theory (MT)
- In RDF MT, an interpretation $I$ of a vocabulary $V$ consists of:
  - $\text{IR}$, a non-empty set of resources
  - $\text{IS}$, a mapping from $V$ into $\text{IR}$
  - $\text{IP}$, a distinguished subset of $\text{IR}$ (the properties)
    - A vocabulary element $v \in V$ is a property iff $\text{IS}(v) \in \text{IP}$
  - $\text{IEXT}$, a mapping from $\text{IP}$ into the powerset of $\text{IR} \times \text{IR}$
    - I.e., a set of elements $\langle x, y \rangle$, with $x, y$ elements of $\text{IR}$
  - $\text{IL}$, a mapping from typed literals into $\text{IR}$
  - Class interpretation $\text{ICEXT}$ simply induced by $\text{IEXT}(\text{IS}(\text{type}))$
    - $\text{ICEXT}(C) = \{ x | \langle x, C \rangle \in \text{IEXT}(\text{IS}(\text{type})) \}$
RDFS Interpretations

- RDFS adds extra constraints on interpretations
  - E.g., interpretations of `<C, subClassOf, D>` constrained to those where $\text{ICEXT}(\text{IS}(C)) \subseteq \text{ICEXT}(\text{IS}(D))$
- Can deal with triples such as
  - `<Species, type, Class>`
  - `<Lion, type, Species>`
  - `<Leo, type, Lion>`
  - `<SelfInst, type, SelfInst>`
- And even with triples such as
  - `<type, subPropertyOf, subClassOf>`
- But not clear if meaning matches intuition (if there is one)
Problems with RDFS

- **RDFS too weak** to describe resources in sufficient detail
  - No **localised range and domain** constraints
    - Can’t say that the range of hasChild is person when applied to persons and elephant when applied to elephants
  - No **existence/cardinality** constraints
    - Can’t say that all instances of person have a mother that is also a person, or that persons have exactly 2 parents
  - No **transitive, inverse or symmetrical** properties
    - Can’t say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
  - ...
- Difficult to provide **reasoning support**
  - No “native” reasoners for non-standard semantics
  - May be possible to reason via FO axiomatisation

Web Ontology Language Requirements

**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
- Formally specified
- Of “adequate” expressive power
- Possible to provide automated reasoning support
From RDF to OWL

- Two languages developed to satisfy above requirements
  - **OIL**: developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - **DAML-ONT**: developed by group of (largely) US researchers (in DARPA DAML programme)
- Efforts merged to produce **DAML+OIL**
  - 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
  - WebOnt group developed **OWL** language based on DAML+OIL
  - OWL language now a W3C Candidate Recommendation
  - Will soon become Proposed Recommendation

OWL Language

- Three species of OWL
  - **OWL full** is union of OWL syntax and RDF
  - **OWL DL** restricted to FOL fragment (∼ DAML+OIL)
  - **OWL Lite** is “easier to implement” subset of OWL DL
- Semantic layering
  - OWL DL ≈ OWL full within DL fragment
  - DL semantics officially definitive
- OWL DL based on **SHIQ** Description Logic
  - In fact it is equivalent to **SHOIN(D)DL**
- OWL DL Benefits from many years of DL research
  - Well defined semantics
  - Formal properties well understood (complexity, decidability)
  - Known reasoning algorithms
  - Implemented systems (highly optimised)
(In)famous “Layer Cake”

- Relationship between layers is not clear
- OWL DL extends “DL subset” of RDF

OWL Class Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>Modal Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \cap \ldots \cap C_n$</td>
<td>Human $\cap$ Male</td>
<td>$C_1 \land \ldots \land C_n$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>Doctor $\cup$ Lawyer</td>
<td>$C_1 \lor \ldots \lor C_n$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \cup \ldots \cup {x_n}$</td>
<td>${\text{john}} \cup {\text{mary}}$</td>
<td>$x_1 \lor \ldots \lor x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$hasChild.Doctor</td>
<td>$(\forall C)\exists C$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$hasChild.Lawyer</td>
<td>$(\exists C)\forall C$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n P$</td>
<td>$\geq 2$hasChild</td>
<td>$(\exists C)^{n+1}$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n P$</td>
<td>$\leq 1$hasChild</td>
<td>$(\forall C)^n$</td>
</tr>
</tbody>
</table>

- XMLS datatypes as well as classes in $\forall P.C$ and $\exists P.C$
  - E.g., $\exists$hasAge.nonNegativeInteger
- Arbitrarily complex nesting of constructors
  - E.g., Person $\forall$ hasChild.Doctor $\exists$hasChild.Doctor
RDFS Syntax

E.g., Person ⊓ ∀ hasChild. Doctor ⊔ ∃ hasChild. Doctor:

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:toClass>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:hasClass rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:toClass>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

OWL Axioms

<table>
<thead>
<tr>
<th>Axiom</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>C₁ ⊑ C₂</td>
<td>Human ⊑ Animal ⊑ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>C₁ ≡ C₂</td>
<td>Man ≡ Human ⊑ Male</td>
</tr>
<tr>
<td>disjointWith</td>
<td>C₁ ⊑ ¬C₂</td>
<td>Male ⊑ ¬Female</td>
</tr>
<tr>
<td>sameIndividualAs</td>
<td>{x₁} ⊑ {x₂}</td>
<td>{President.Bush} ≡ {G.W.Bush}</td>
</tr>
<tr>
<td>differentFrom</td>
<td>{x₁} ⊑ ¬{x₂}</td>
<td>{john} ⊑ ¬{peter}</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>P₁ ⊑ P₂</td>
<td>hasDaughter ⊑ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>P₁ ≡ P₂</td>
<td>cost ≡ price</td>
</tr>
<tr>
<td>inverseOf</td>
<td>P₁ ⊑ P₂</td>
<td>hasChild ≡ hasParent⁻</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>P⁺ ⊑ P</td>
<td>ancestor⁺ ⊑ ancestor⁻</td>
</tr>
<tr>
<td>functionalProperty</td>
<td>T ⊑ 1P</td>
<td>T ⊑ 1hasMother</td>
</tr>
<tr>
<td>inverseFunctionalProperty</td>
<td>T ⊑ 1P⁻</td>
<td>T ⊑ 1hasSSN⁻</td>
</tr>
</tbody>
</table>

- **Axioms (mostly) reducible to inclusion ( )**
  - C ⇒ D if both C ⊑ D and D ⊑ C
XML Schema Datatypes in OWL

- OWL supports XML Schema primitive datatypes
  - E.g., integer, real, string, …
- Strict separation between “object” classes and datatypes
  - Disjoint interpretation domain $\Delta_d$ for datatypes
    - For a datavalue $d$, $d \subseteq \Delta_d$
    - And $\Delta_o \cap \Delta_d = \emptyset$
  - Disjoint “object” and datatype properties
    - For a datatype property $P$, $P \subseteq \Delta_i \times \Delta_d$
    - For object property $S$ and datatype property $P$, $S \cap P = \emptyset$
- Equivalent to the “$\langle D_n \rangle$” in $\text{SHOIN}(D_n)$

Why Separate Classes and Datatypes?

- Philosophical reasons:
  - Datatypes structured by built-in predicates
  - Not appropriate to form new datatypes using ontology language
- Practical reasons:
  - Ontology language remains simple and compact
  - Semantic integrity of ontology language not compromised
  - Implementability not compromised — can use hybrid reasoner
    - Only need sound and complete decision procedure for:
      $d_1 \cap \ldots \cap d_n$, where $d$ is a (possibly negated) datatype
OWL DL Semantics

- Mapping OWL to equivalent DL (SHOIN-DL):
  - Facilitates provision of reasoning services (using DL systems)
  - Provides well defined semantics
- DL semantics defined by interpretations: \( I = (\Delta^I, \cdot^I) \), where
  - \( \Delta^I \) is the domain (a non-empty set)
  - \( \cdot^I \) is an interpretation function that maps:
    - Concept (class) name \( A \rightarrow \) subset \( A^I \) of \( \Delta^I \)
    - Role (property) name \( R \rightarrow \) binary relation \( R^I \) over \( \Delta^I \)
    - Individual name \( i \rightarrow i^I \) element of \( \Delta^I \)

DL Semantics

- Interpretation function \( \cdot^I \) extends to concept expressions in an obvious(ish) way, i.e.:

  
  \[
  \begin{align*}
  (C \cap D)^I & = C^I \cap D^I \\
  (C \cup D)^I & = C^I \cup D^I \\
  (\neg C)^I & = \Delta^I \setminus C^I \\
  \{x\}^I & = \{x^I\} \\
  (\exists R.C)^I & = \{x \mid \exists y.(x,y) \in R^I \land y \in C^I\} \\
  (\forall R.C)^I & = \{x \mid \forall y.(x,y) \in R^I \Rightarrow y \in C^I\} \\
  (\leq n R)^I & = \{x \mid \# \{y \mid \langle x,y \rangle \in R^I\} \leq n\} \\
  (\geq n R)^I & = \{x \mid \# \{y \mid \langle x,y \rangle \in R^I\} \geq n\}
  \end{align*}
  \]
DL Knowledge Bases (Ontologies)

- An OWL ontology maps to a DL Knowledge Base $K = T, A$
  - $T$ (Tbox) is a set of axioms of the form:
    - $C \sqsubseteq D$ (concept inclusion)
    - $C \equiv D$ (concept equivalence)
    - $R \sqsubseteq S$ (role inclusion)
    - $R \equiv S$ (role equivalence)
    - $R^+ \sqsubseteq R$ (role transitivity)
  - $A$ (Abox) is a set of axioms of the form
    - $x \in D$ (concept instantiation)
    - $\langle x, y \rangle \in R$ (role instantiation)
- Two sorts of Tbox axioms often distinguished
  - “Definitions”
    - $C \sqsubseteq D$ or $C \equiv D$ where $C$ is a concept name
  - General Concept Inclusion axioms (GCIs)
    - $C \sqsubseteq D$ where $C$ is an arbitrary concept

Knowledge Base Semantics

- An interpretation $I$ satisfies (models) an axiom $A$ ($I \models A$):
  - $I \models C \sqsubseteq D$
  - $I \models C \equiv D$
  - $I \models R \sqsubseteq S$
  - $I \models x \in D$
  - $I \models \langle x, y \rangle \in R$
- $I$ satisfies a Tbox $T$ ($I \models T$) iff $I$ satisfies every axiom $A$ in $T$
- $I$ satisfies an Abox $A$ ($I \models A$) iff $I$ satisfies every axiom $A$ in $A$
- $I$ satisfies a KB $K$ ($I \models K$) iff $I$ satisfies both $T$ and $A$
Inference Tasks

• Knowledge is correct (captures intuitions)
  – \( C \) subsumes \( D \) w.r.t. \( K \) iff for every model \( I \) of \( K \), \( C^I \subseteq D^I \)

• Knowledge is minimally redundant (no unintended synonyms)
  – \( C \) is equivalent to \( D \) w.r.t. \( K \) iff for every model \( I \) of \( K \), \( C^I = D^I \)

• Knowledge is meaningful (classes can have instances)
  – \( C \) is satisfiable w.r.t. \( K \) iff there exists some model \( I \) of \( K \) s.t. \( C^I \neq \emptyset \)

• Querying knowledge
  – \( x \) is an instance of \( C \) w.r.t. \( K \) iff for every model \( I \) of \( K \), \( x^I \in C^I \)
  – \( \langle x, y \rangle \) is an instance of \( R \) w.r.t. \( K \) iff for every model \( I \) of \( K \), \( (x^I , y^I) \in R^I \)

• Knowledge base consistency
  – A KB \( K \) is consistent iff there exists some model \( I \) of \( K \)

Acknowledgements

Thanks to various people from whom I “borrowed” material:

– Jeen Broekstra
– Carole Goble
– Frank van Harmelen
– Austin Tate
– Raphael Volz

And thanks to all the people from whom they borrowed it ☺