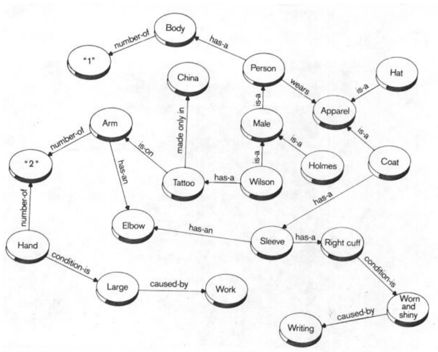
Ontology Languages
for the
Semantic Web

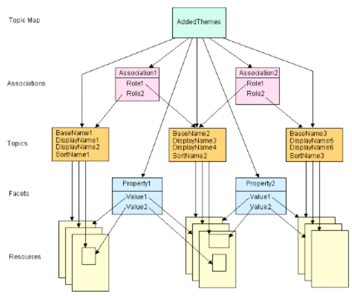
Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - Semantic networks



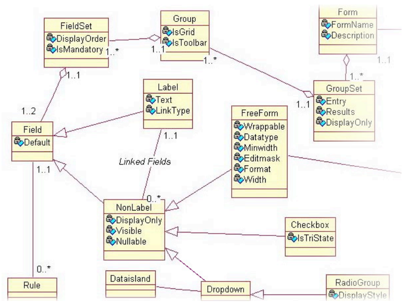
Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - Topic Maps



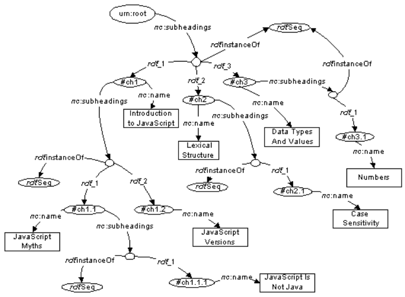
Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - UML



Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Graphical notations
    - RDF



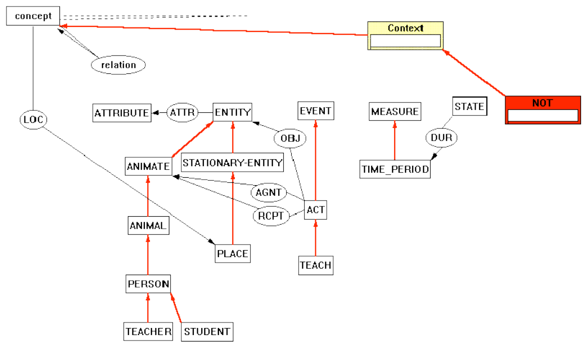
Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Logic based
    - Description Logics (e.g., OIL, DAML+OIL, OWL)
    - Rules (e.g., RuleML, LP/Prolog)
    - First Order Logic (e.g., KIF)

Every gardener likes the sun.
(Ax) gardener(x) => likes(x,Sun)
You can fool some of the people all of the time.
(Ex)(At) (person(x) ^ time(t)) => can-fool(x,t)
You can fool all of the people some of the time.
(Ax)(Et) (person(x) ^ time(t) => can-fool(x,t)
All purple mushrooms are poisonous.
(Ax) (mushroom(x) ^ purple(x)) => poisonous(x)
No purple mushroom is poisonous.
~(Ex) purple(x) ^ mushroom(x) ^ poisonous(x)
(Ax) (mushroom(x) ^ purple(x)) => ~poisonous(x)
There are exactly two purple mushrooms.
(Ex)(Ey) mushroom(x) ^ purple(x) ^ mushroom(y) ^ purple(y) ^ ~(x=y) ^ (Az) (mushroom(z) ^ purple(z)) => ((x=z) v (y=z))
Clinton is not tall.
~tall(Clinton)

Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Logic based
    - Conceptual graphs



Ontology Languages

- Wide variety of languages for “Explicit Specification”
  - Logic based
    - Conceptual graphs
    - (Syntactically) higher order logics (e.g., LBase)
    - Non-classical logics (e.g., Flogic, Non-Mon, modalities)
  - Bayesian/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

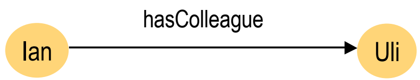
- Existing Web languages extended to facilitate content description
  - XML → XML Schema (XMLS)
  - RDF → RDF Schema (RDFS)
- XMLS *not* an ontology language
  - Changes format of DTDs (document schemas) to be XML
  - Adds an extensible type hierarchy
    - Integers, Strings, etc.
    - Can define sub-types, 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:



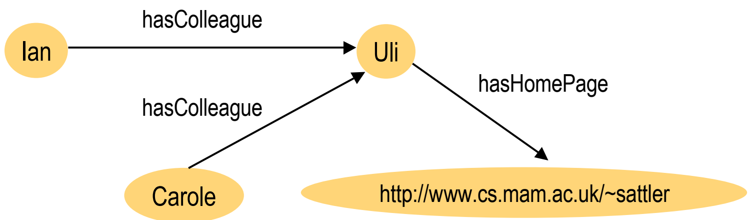
- Can be represented using XML serialisation, e.g.:
  `<Ian,hasColleague,Uli>`
- Statements describe properties of resources
- A resource is a URI representing a (class of) object(s):
  - a document, a picture, a paragraph on the Web;
  - http://www.cs.man.ac.uk/index.html
  - a book in the library, 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“
- URIs may or may not be dereferencable
  - 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



- 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 with an associated object resource
- Resources are referred to using URIs

```
<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
  - I.e., <Person,subClassOf,Animal> has no special meaning

- RDF Schema defines "schema vocabulary" that supports definition of ontologies
  - gives "extra meaning" to particular RDF predicates and resources (such as subClasOf)
  - 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)

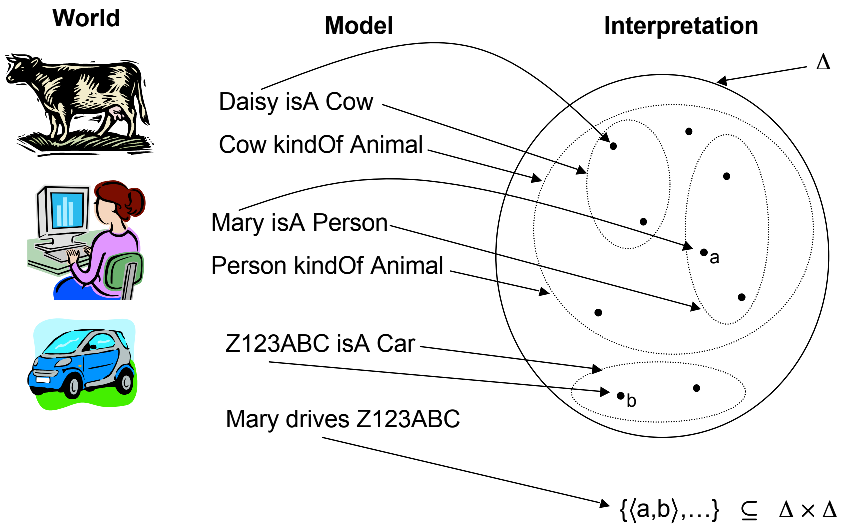
Aside: Semantics and Model Theories

- 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., $\mathcal{T} \models A \sqsubseteq B$ iff in every model of $\mathcal{T}$, $ext(A) \subseteq ext(B)$

Aside: Set Based Model Theory

- Many logics (including standard First Order Logic) use a model theory based on Zermelo-Frankel set theory
- The domain of discourse (i.e., the part of the world being modelled) is represented as a set (often refered as Δ)
- Objects in the world are interpreted as elements of Δ
  - Classes/concepts (unary predicates) are subsets of Δ
  - Properties/roles (binary predicates) are subsets of $\Delta \times \Delta$ (i.e., Δ^2)
  - Ternary predicates are subsets of Δ^3 etc.
- The sub-class relationship between classes can be interpreted as set inclusion
- Doesn't work for RDF, because in RDF a class (set) can be a member (element) of another class (set)
  - In Z-F set theory, elements of classes are atomic (no structure)

Aside: Set Based Model Theory Example



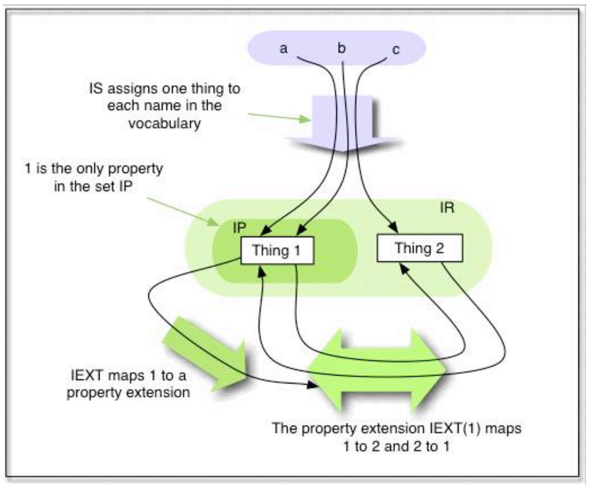
Aside: Set Based Model Theory Example

- **Formally, the vocabulary is the set of names we use in our model of (part of) the world**
  - **{Daisy, Cow, Animal, Mary, Person, Z123ABC, Car, drives, …}**
- **An interpretation $\mathcal{I}$ is a tuple $\langle \Delta, \cdot^{\mathcal{I}} \rangle$**
  - **Δ is the domain (a set)**
  - **$\cdot^{\mathcal{I}}$ is a mapping that maps**
    - **Names of objects to elements of Δ**
    - **Names of unary predicates (classes/concepts) to subsets of Δ**
    - **Names of binary predicates (properties/roles) to subsets of $\Delta \times \Delta$**
    - **And so on for higher arity predicates (if any)**

RDF Semantics

- **RDF has "Non-standard" semantics in order to deal with this**
- **Semantics given by RDF Model Theory (MT)**
- **In RDF MT, an interpretation $\mathcal{I}$ of a vocabulary V consists of:**
  - **IR, a non-empty set of resources (corresponds to Δ)**
  - **IS, a mapping from V into IR (corresponds to $\cdot^{\mathcal{I}}$)**
  - **IP, a distinguished subset of IR (the properties)**
    - **A vocabulary element $v \in V$ is a property iff $IS(v) \in IP$**
  - **IEXT, a mapping from IP into the powerset of IR×IR**
    - **I.e., property elements mapped to subsets of IR×IR**
  - **IL, a mapping from typed literals into IR**

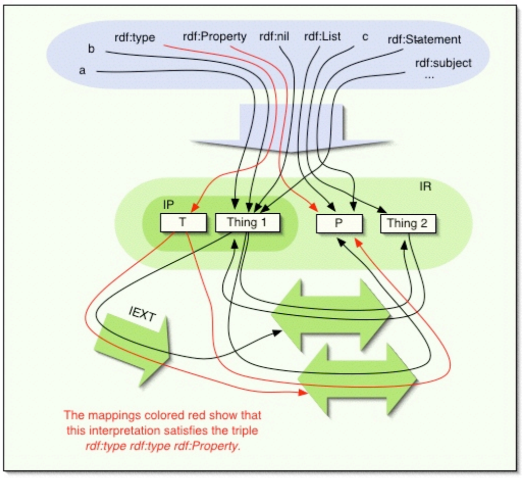
Example RDF Simple Interpretation



RDF Semantic Conditions

- RDF Imposes **semantic conditions** on interpretations, e.g.:
  - x is in IP if and only if <x, IS(rdf:Property)> is in IEXT(I(rdf:type))
- All RDF interpretations must satisfy certain **axiomatic triples**, e.g.:
  - rdf:type rdf:type rdf:Property
  - rdf:subject rdf:type rdf:Property
  - rdf:predicate rdf:type rdf:Property
  - rdf:object rdf:type rdf:Property
  - rdf:first rdf:type rdf:Property
  - rdf:rest rdf:type rdf:Property
  - rdf:value rdf:type rdf:Property
  - …

Example RDF Interpretation



RDFS Semantics

- RDFS simply adds semantic conditions and axiomatic triples that give meaning to schema vocabulary
- Class interpretation ICEXT simply induced by rdf:type, i.e.:
  - x is in ICEXT(y) if and only if <x,y> is in IEXT(IS(rdf:type))
- Other semantic conditions include:
  - If <x,y> is in IEXT(IS(rdfs:domain)) and <u,v> is in IEXT(x) then u is in ICEXT(y)
  - If <x,y> is in IEXT(IS(rdfs:subClassOf)) then x and y are in IC and ICEXT(x) is a subset of ICEXT(y)
  - IEXT(IS(rdfs:subClassOf)) is transitive and reflexive on IC
- Axiomatic triples include:
  - rdf:type rdfs:domain rdfs:Resource
  - rdfs:domain rdfs:domain rdf:Property

RDFS Interpretation Example

- If RDFS graph includes triples

  <Species,type,Class>
  <Lion,type,Species>
  <Leo,type,Lion>
  <Lion,subClassOf,Mamal>
  <Mamal,subClassOf,Animal>

- Interpretation conditions imply existence of triples

  <Lion,subClassOf,Animal>
  <Leo,type,Mamal>
  <Leo,type,Animal>
  …

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

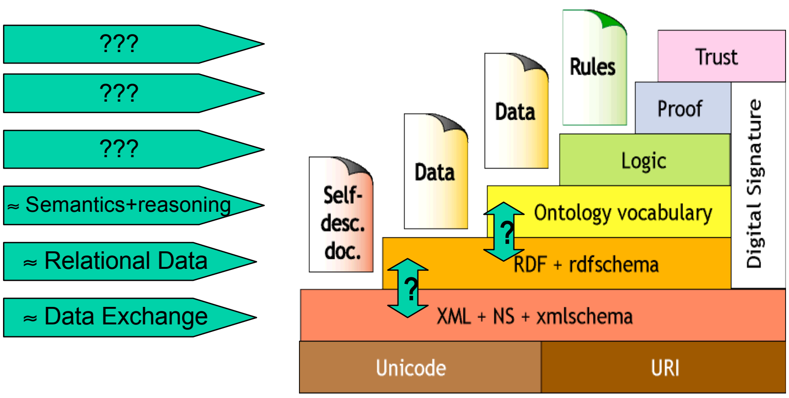
From RDF to OWL

- 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 Recommendation (i.e., a standard like HTML and XML)

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 $\mathcal{SHIQ}$ Description Logic
  - In fact it is equivalent to $\mathcal{SHOIN}(D_n)$ 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