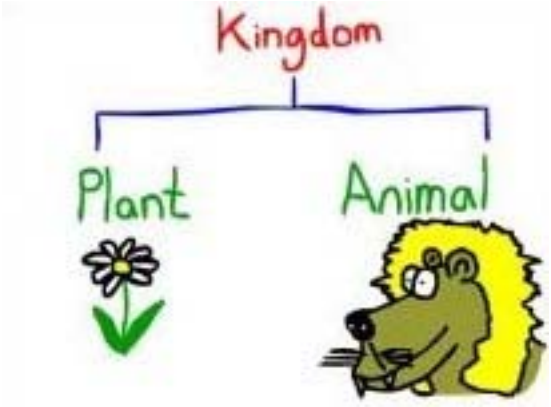
MANCHESTER 1824

The University of Manchester



Ontology Engineering: Tools and Methodologies

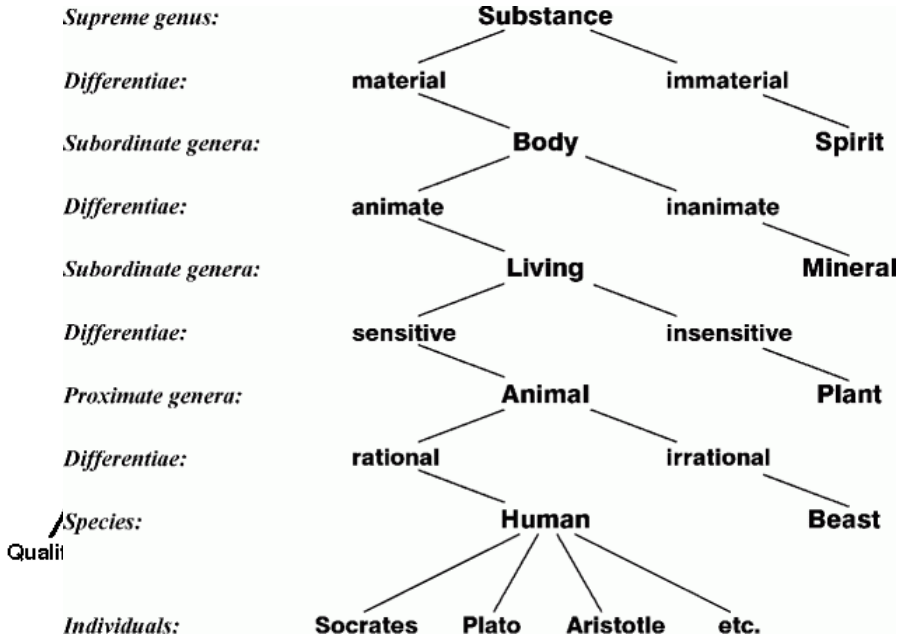
Ian Horrocks
<horrocks@cs.man.ac.uk>
Information Management Group
School of Computer Science
University of Manchester

iM.g

Ontologies

Ontology: Origins and History

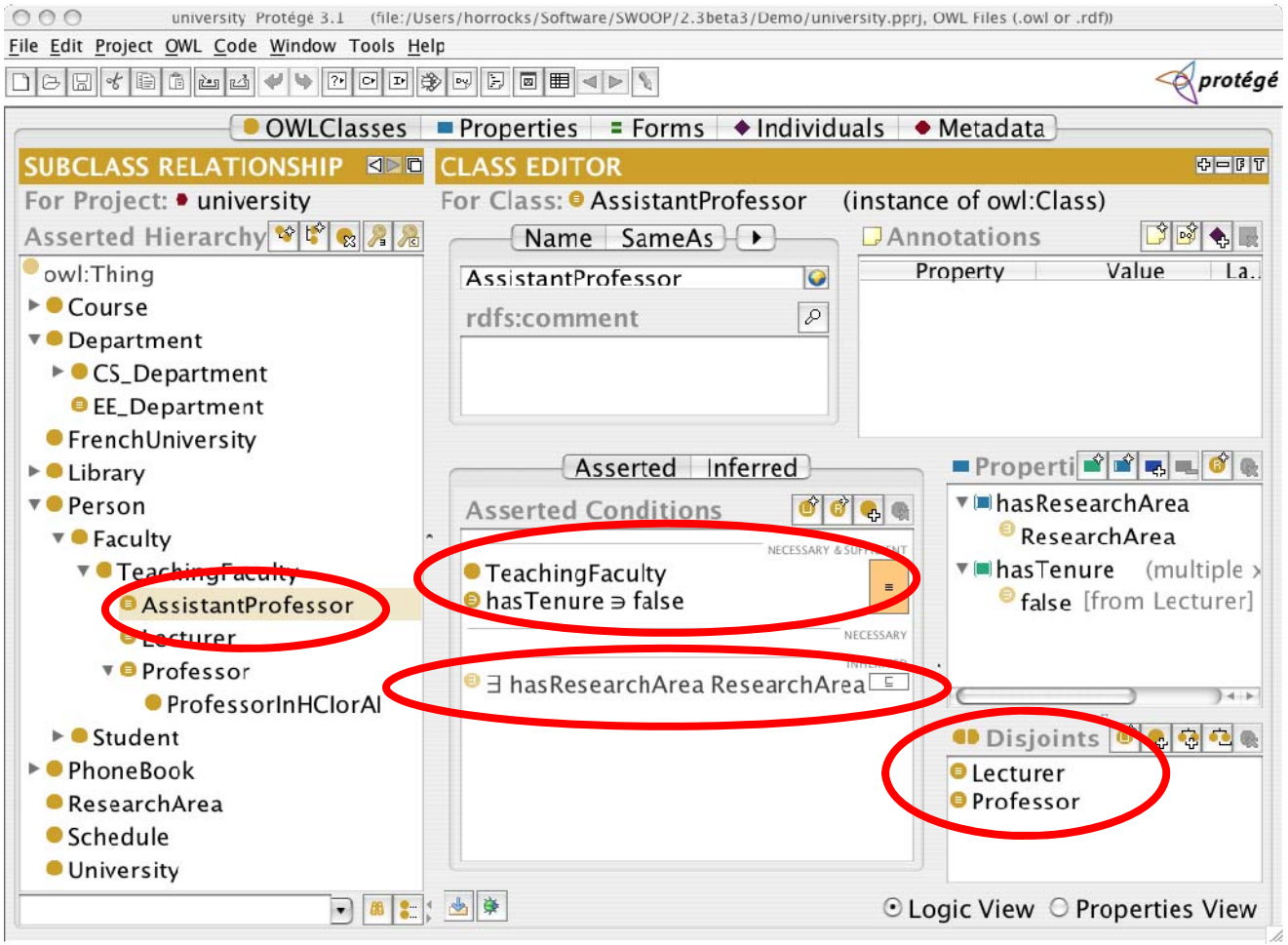
- In Philosophy, fundamental branch of metaphysics
  - Studies “being” or “existence” and their **basic categories**
  - Aims to find out what **entities** and **types of entities** exist



Ontology in Information Science

- An ontology is an engineering artefact consisting of:
  - A **vocabulary** used to describe (a particular view of) some domain
  - An **explicit specification** of the **intended meaning** of the vocabulary.
    - Often includes classification based information
  - Constraints capturing **background knowledge** about the domain
- Ideally, an ontology should:
  - Capture a **shared understanding** of a domain of interest
  - Provide a **formal** and **machine manipulable** model

Example Ontology (Protégé)



The Web Ontology Language OWL

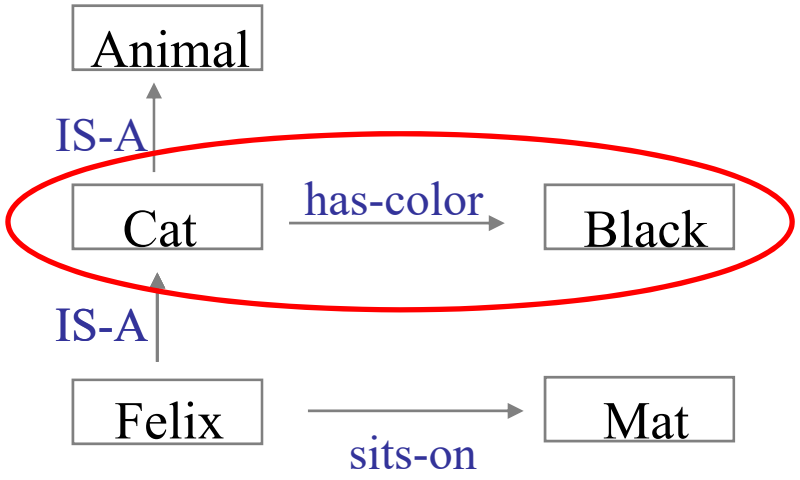
What Are Description Logics?

- A family of logic based Knowledge Representation formalisms
  - Descendants of **semantic networks** and **KL-ONE**
  - Describe domain in terms of **concepts** (classes), **roles** (properties, relationships) and **individuals**
  - **Operators** allow for composition of complex concepts
  - **Names** can be given to complex concepts, e.g.:

HappyParent ´ Parent u 8hasChild.(Intelligent t Athletic)

Why (Description) Logic?

- OWL exploits results of 15+ years of DL research
  - Well defined (model theoretic) **semantics**



[Quillian, 1967]

Why (Description) Logic?

- OWL exploits results of 15+ years of DL research
  - Well defined (model theoretic) **semantics**
  - **Formal properties** well understood (complexity, decidability)
  - Known **reasoning** **algorithms**
  - **Implemented systems** (highly optimised)

Pellet

KAON2

CEL

Why the Strange Names?

- Description Logics are a **family** of KR formalisms
  - Mainly distinguished by available operators
- **Available operators** indicated by letters in name, e.g.,
  - **S** : basic DL (ALC) plus transitive roles (e.g., ancestor $\in R_+$)
  - **H** : role hierarchy (e.g., hasDaughter **v** hasChild)
  - **O** : nominals/singleton classes (e.g., {Italy})
  - **I** : inverse roles (e.g., isChildOf ´ hasChild$^-$)
  - **N** : number restrictions (e.g., **>**2hasChild, **6**3hasChild)
- Basic DL + role hierarchy + nominals + inverse + NR = **SHOIN**
  - The basis for **OWL-DL**
- SHOIN is very expressive, but still decidable (just)
  - Decidable $\Rightarrow$ we can build reliable tools and reasoners

Why (Description) Logic?

- Foundational research was **crucial** to design of OWL
  - Informed Working Group decisions at every stage, e.g.:
    - "Why not extend the language with feature **x**, which is clearly harmless?"

    - "Adding **x** would lead to undecidability - see proof in […]"

Class/Concept Constructors

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

- C is a concept (class); P is a role (property); x is an individual name
- XMLS datatypes as well as classes in 8P.C and 9P.C
  - Restricted form of DL concrete domains

Knowledge Base / Ontology Axioms

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	$\langle$John, Mary$\rangle$: has-child

Knowledge Base / Ontology

- A TBox is a set of "schema" axioms (sentences), e.g.:

$$\{\text{Parent} \sqsubseteq \text{Person} \sqcap \geq 1\,\text{hasChild},\ \text{HappyParent} \doteq \text{Parent} \sqcap \forall\,\text{hasChild}.(\text{Intelligent} \sqcup \text{Athletic})\}$$

- An ABox is a set of "data" axioms (ground facts), e.g.:

$$\{\text{John}:\text{HappyParent},\ \langle \text{John}, \text{Mary}\rangle:\text{hasChild}\}$$

- An OWL ontology is just a **SHOIN** KB

Why Ontology Reasoning?

- Given key role of ontologies in many applications, it is 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** — captures intuitions of domain experts
    - **Minimally redundant** — no unintended synonyms
  - Answer **queries**, e.g.:
    - Find more general/specific classes
    - Retrieve individuals/tuples matching a given query