

# Modularity in Ontologies: Introduction (Part A)

Thomas Schneider<sup>1</sup>    *Dirk Walther*<sup>2</sup>

<sup>1</sup>Department of Computer Science, University of Bremen, Germany

<sup>2</sup>Faculty of Informatics, Technical University of Madrid, Spain

ESLLI, 1 August 2011



# Course Objective

- Ontologies are widely used as means to represent conceptualisations within a domain. In Computer Science, in particular, ontologies mainly provide a reference vocabulary for a domain.
- Many ontologies have been developed in several areas broad and diverse such as life sciences, health-care, linguistics, geosciences, etc.
- Examples: SNOMED CT, FMA, GALEN, GO, NCI, etc.
- We regard ontologies as logical theories.
- Challenge: Provide automatic support for sharing and reuse of (large) ontologies as well as their design and maintenance.
- Modularity is a key concept in tackling the challenges.
- We will define the notion of modularity for ontologies and provide an overview on its properties and uses as well as an introduction to related technical results.



# (Tentative) Course Outline

- Monday: introduction (Thomas and Dirk)
- Tuesday: formal foundations of modularity (Dirk)
- Wednesday: module extraction (Thomas)
- Thursday: ontology versioning, forgetting of vocabulary (Dirk)
- Friday: recent advances/current work (Thomas)

## Prerequisites:

- basic understanding of Description Logic *or* related logic formalisms
- basic knowledge of complexity theory



General goal of KR: “develop formalisms for providing high-level descriptions of the world that can be effectively used to build intelligent applications” [Brachman and Nardi, 2003]

Requirements to a KR system:

- well-defined syntax and unambiguous semantics (machine processable)
- appropriate abstraction level (relevant vs. irrelevant aspects)
- reasoning about represented knowledge (esp. drawing of inferences of implicit from explicit knowledge)
- practical reasoning tools



Early KR approaches:

- Semantic Networks [Quillian, 1967]
- Frame Systems [Minsky, 1981]

⇒ problem: no formal semantics

Logical formalisms:

- Logics have a formal syntax and semantics.
- Various logics with different expressivity are available.
- There are reasoning algorithms for many decidable logics.
- Optimised reasoning systems are available.



- Logic = formal language  $L$  + formal semantics ' $\models$ '
- **Logical theory**: a set  $T$  of  $L$ -formulas (theorems) closed under logical consequences

if  $T \models \varphi$ , then  $\varphi \in T$

- Here we mostly consider finite axiomatisations of a theory and call these **ontologies**.



# Examples of FOL Theories

- theory of a functional relation
  - signature: unary function symbol  $f(\cdot)$
  - axiom:  $\forall x \forall y ((f(x) = f(y)) \rightarrow (x = y))$
- theory of natural numbers
  - signature: constant  $0$ , unary function symbol  $s(\cdot)$
  - axioms:  $\forall x \neg (s(x) = 0)$ ,  $\forall x \forall y ((s(x) = s(y)) \rightarrow (x = y))$
- theory of Boolean algebras ...
- theory of a structure  $N = (\mathbb{N}, 0, s, +)$ 
  - signature: constant  $0$ , unary function symbol  $s(\cdot)$ , binary function symbol  $+(\cdot, \cdot)$
  - theory consists of the FO sentences  $\varphi$  of that signature with  $N \models \varphi$



- propositional logic: constraint satisfaction problems, planning
- first-order logic: specification of graphs, datatypes
- temporal logic: specification of hard- and software systems
- epistemic logic: specification of agents' knowledge and belief
- non-monotonic logic: default reasoning, abductive reasoning, belief revision
- description logic: specification of vocabulary in an **ontology** (terminology)





In Computer/Information Science, ontologies are “a formal, explicit specification of a shared conceptualisation” [Gruber, 1993]

An *ontology*:

- covers a *domain* of interest, e.g., medicine, biology, geography, linguistics, ESSLLI, etc.
- presents a *model* of the domain
- provides a *vocabulary* with names for objects, relations and classes and defines relationships between them
- can define names for terms from existing ones
- is presented in a formal language with formal semantics
- is shared among users



Examples of vocabulary classifications (taxonomies):

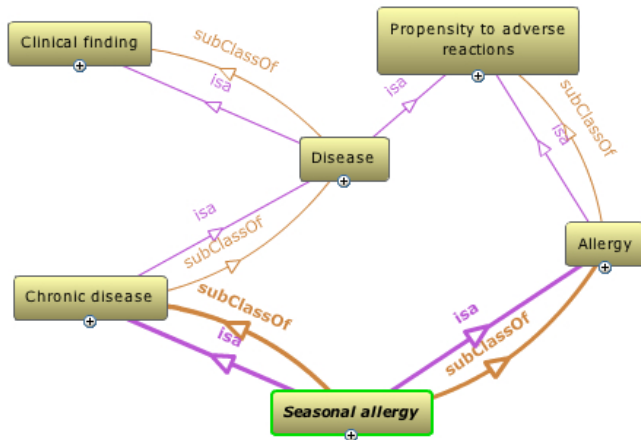
- astronomy: Secchi classes to classify stars (Secchi, 1877)
- biology: Linnaean taxonomy (Linnaeus, 1735) classifying organisms
- gastronomy: Bordeaux Wine Official Classification of 1855
- libraries: Dewey Decimal Classification (Dewey, 1876)
- medicine: International Statistical Classification of Diseases and Related Health Problems (ICD)
- common sense knowledge: OpenCyc ([opencyc.org](http://opencyc.org))
- websites: Open Directory Project ([dmoz.org](http://dmoz.org))



Examples of ontologies in medicine and biology:

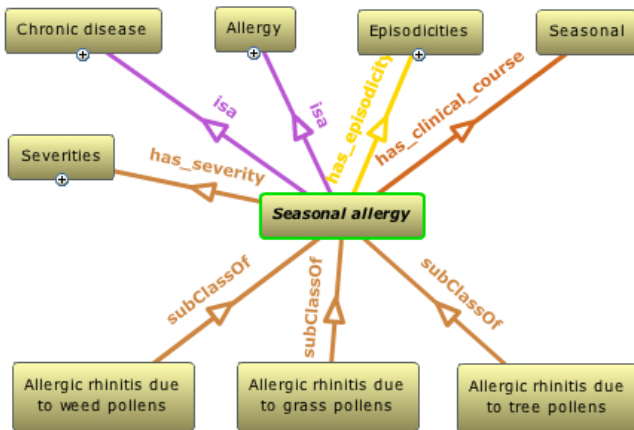
- Gene Ontology ([geneontology.org](http://geneontology.org))
  - provides vocabularies for the annotation of gene products
  - `protein_tag`  $\sqsubseteq$  `molecular_function`
- MGED Ontology ([mged.sourceforge.net/ontologies/](http://mged.sourceforge.net/ontologies/))
  - standardising terms for the annotation of microarray experiments
  - `DiseaseState`  $\sqsubseteq$  `BioMaterialCharacteristics`
- National Cancer Institutes Thesaurus ([ncicb.nci.nih.gov](http://ncicb.nci.nih.gov))
  - standardising terms for the annotation of microarray experiments
  - `HIV_Budding`  $\sqsubseteq$  `Virus-Cell_Membrane_Interaction`
- Systematized Nomenclature of Medicine Clinical Terms ([ihtsdo.org](http://ihtsdo.org))
  - provides medical terminology
  - `Seasonal_Allergy`  $\sqsubseteq$  `Chronic_disease`





- class hierarchy (taxonomy) from Seasonal\_Allergy (using [bioportal.bioontology.org](http://bioportal.bioontology.org))





- neighbourhood of Seasonal\_Allergy (using `bioportal.bioontology.org`)

- “provides a common framework that allows data to be **shared** and **reused** across application, enterprise, and community boundaries” [W3C, 2010]
- extends World Wide Web with meta data (e.g. annotations) about the pages and how they relate to each other
- goal: machines automatically process information on the web (find, share, combine, act upon, reason with information, etc.)  
⇒ “intelligent machines”
- name coined by Tim Berners-Lee
- some functionality:
  - answer queries involving background knowledge
  - access information in data repositories
  - use web services
  - delegate tasks to agents



# Examples of Ontology languages

- formalism for knowledge representation
  - ER- and UML diagrams
  - Conceptual Graphs
  - Datalog and rule-based languages
  - DLs and higher-order logics
- traditional ontology specification languages
  - Ontolingua
  - Operational Conceptual Modeling Language (OCML)
  - Frame Logic
- web standards and W3C recommendations
  - eXtended Markup Language (XML)
  - Resource Description Framework (RDF) and RDF Schema
  - Web Ontology Language (OWL)



# Web Ontology Language (OWL)

- ontology language for the Semantic Web with formally defined meaning
- designed to facilitate ontology development and sharing via the Web
- provide **classes**, **properties**, **individuals**, and **data values**
- a standard for ontologies in applications in the web (also used independently of the web)
- RDF/XML-based syntax
- W3C standards: OWL (2004), OWL 2 (2009)  
(technical reports available under [www.w3.org/TR/](http://www.w3.org/TR/))
- profiles (sub-languages) to trade expressive power for performance guarantees of reasoning





- OWL Full:
  - very expressive
  - no strict separation of classes, properties, individuals and data values (e.g. allows classes to be treated as individuals)
  - undecidable (no complete reasoning)
- OWL DL:
  - restricts OWL Full to obtain computational guarantees
  - optimised reasoning systems available
  - based on **SHOIN** with XML datatypes
- OWL Lite
  - provides a minimal useful subset of language features that are easy to implement
  - based on **SHIN** with XML datatypes
  - reasoning not tractable



- OWL 2:
  - backward-compatible to OWL 1
  - based on **SROIQ** with XML datatypes
  - extension of OWL DL to provide the ontology developer with any desirable (but reasonable) expressive means for easy and intuitive modelling
  - high reasoning complexity; possibly acceptable in practice (?)
  - reasoning systems available
- OWL 2 EL:
  - suitable for large-scale ontologies of restricted expressivity
  - based on **EL**-family of DLs
  - tractable reasoning



- OWL 2 QL:
  - query language for answering conjunctive queries in LogSpace using relational database technology
  - suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to access the data directly via relational queries (e.g., SQL)
  - similar to DL *DL-Lite<sub>horn</sub>* - captures most of ER- and UML diagrams
- OWL 2 RL:
  - enables the implementation of polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples
  - it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples
  - inspired by DL programs



- “family of logic-based knowledge representation formalisms”  
⇒ fulfill requirements to a KR system
- W3C recommends to base OWL languages onto DL
- expressivity vs. computational complexity
- DLs define **classes**, **properties/relations** and **objects** using **concepts**, **roles** and **individuals**
- concept language of DLs:
  - *concept names* are names for groups of objects
  - *role names* are names for relations between objects
  - *individual names* are names for objects
  - *constructors* relate names for concepts, roles and individuals



# Example: a terminology of ESSLLI

- classes (concepts): Person, Course, Lecturer, Attendant, ...
- relations (roles): attends, gives, likes, ...
- objects (individuals): Thomas, x, y, ...
- definitions:
  - $\text{Lecturer} \equiv \text{Person} \sqcap \exists \text{gives.Course}$
  - $\text{Attendant} \equiv \text{Person} \sqcap \exists \text{attends.Course}$
  - $\text{Registrant} \equiv \text{Person} \sqcap \text{Registered}$
- assertions:
  - $\text{Lecturer}(\text{Thomas}), \text{Attendant}(x), \text{Attendant}(y)$
  - $\text{gives}(\text{Thomas}, \text{mod-course}), \text{likes}(x, \text{mod-course}), \text{likes}(x, y)$
- constraints:
  - $\text{Workshop} \sqsubseteq \forall \text{attended\_by.Registrant}$
  - $\text{attended\_by} \equiv \text{attends}^{-1}$



- DLs can be embedded into FOL
  - concepts correspond to unary predicates
  - roles correspond to binary predicates
  - no more than 2 variables under the scope of a quantifier  
(exception: transitive roles, number restrictions, etc.)
  - individuals correspond to constants
  - no function symbols
- DLs are usually decidable



# The basic Description Logic ALC

- signature: countably infinite supply of concept names  $A, B, \dots$ , role names  $r, s, \dots$  and individual names  $a, b, \dots$
- syntax:

$$C, D ::= \top \mid \perp \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.D$$

- individual assertions:
  - $C(a)$
  - $r(a, b)$
- axioms:
  - $C \sqsubseteq D$
  - $C \equiv D$
- ABox: finite set of individual assertions
- TBox: finite set of axioms



# ALC Interpretations

- interpretation  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ 
  - $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$  for all concept names  $A$
  - $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$  for all role names  $r$

Name	Syntax	Semantics
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\perp$	$\emptyset$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in C^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}}\}$
universal restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in C^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}}\}$





An interpretation  $\mathcal{I}$  satisfies:

- concept inclusion:  $C \sqsubseteq D$  iff  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- concept equation:  $C \equiv D$  iff  $C^{\mathcal{I}} = D^{\mathcal{I}}$
- TBox:  $T$  iff  $\mathcal{I}$  satisfies all axioms in  $T$  ( $\mathcal{I}$  is a model of  $T$ )
- concept assertion:  $C(a)$  iff  $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- role assertion:  $r(a, b)$  iff  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$
- ABox:  $A$  iff  $\mathcal{I}$  satisfies all assertions in  $A$  ( $\mathcal{I}$  is a model of  $A$ )



# Common Reasoning Tasks

- (1) Subsumption of concepts  $C, D$  wrt. TBox  $T$ 
  - Does  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$  hold in all models of  $T$ ?
- (2) Satisfiability of concept  $C$  wrt. TBox  $T$ 
  - Is there a model  $\mathcal{I}$  of  $T$  such that  $C^{\mathcal{I}} \neq \emptyset$ ?
- (3) Consistency of KB  $K = (T, A)$ 
  - Is there a common model of  $T$  and  $A$ ?
- (4) Instance checking of individual  $a$  in concept  $C$  wrt. KB  $K = (T, A)$ 
  - Does  $a^{\mathcal{I}} \in C^{\mathcal{I}}$  hold in all models  $\mathcal{I}$  of  $K$ ?
- (5) Query answering
  - Given a KB  $K = (T, A)$ , a query  $q(\vec{x})$  and a tuple  $\vec{a}$  of individual names from  $A$ , does  $\mathcal{I}$  satisfy  $q(\vec{a})$  for all models  $\mathcal{I}$  of  $K$ ?



- provide tractable reasoning
- **DL-Lite** family [Calvanese et al., 2007]
  - conceptual modelling (capture much of ER- and UML-diagrams)
  - designed to access large amounts of data via high-level conceptual interface (data integration, querying instance data using background theories)
- **EL** family [Baader, Brandt, Lutz, 2005]
  - captures large biomedical ontologies like SNOMED CT, NCI thesaurus
- **common restrictions: no disjunction, no universal restriction**



- extension of *SHOIN* [Horrocks, Kutz, Sattler, 2006]
- provides the ontology developer with any desirable (but reasonable) expressive means for easy and intuitive modelling
- reasoning is 2NExpTime-complete [Kazakov, 2008]
- *ALC* extended with:
  - nominals
  - qualified number restrictions
  - conditions on roles: (ir)reflexivity, symmetry, transitivity and universality
  - conditions between roles: complex role inclusions and disjointness



# Extending ALC to SROIQ

- syntax (additional to ALC):

$$C, D ::= \{a\} \mid \exists s.\text{self} \mid (\leq n s C) \mid (\geq n s C)$$

where  $s$  is a **simple** role

- simple roles:
  - necessary restriction on roles to gain decidability
  - intuitively:  $s$  is simple if  $s$  is not implied by compositions of roles
- individual assertions:
  - $\neg s(a, b)$  (also  $(a, b) : \neg s$ )
  - $a \neq b$
- axioms:
  - role hierarchy:  $r \sqsubseteq r'$
  - transitive roles:  $r \circ r \sqsubseteq r$
- RBox: finite set of role inclusion axioms and role assertions



Name	Syntax	Semantics
$\exists s.\text{self-concepts}$	$\exists s.\text{self}$	$\{x \mid (x, x) \in s^{\mathcal{I}}\}$
at-least restriction	$(\geq n \text{ } s \text{ } C)$	$\{x \mid \#\{y \mid (x, y) \in r^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \geq n\}$
at-most restriction	$(\leq n \text{ } s \text{ } C)$	$\{x \mid \#\{y \mid (x, y) \in r^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \leq n\}$

An interpretation  $\mathcal{I}$  satisfies:

- inequality assertion:  $a \neq b$  iff  $a^{\mathcal{I}} \neq b^{\mathcal{I}}$
- negated role assertion:  $\neg r(a, b)$  iff  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$
- RBox:  $R$  iff  $\mathcal{I}$  satisfies all assertions in  $R$  ( $\mathcal{I}$  is a model of  $R$ )

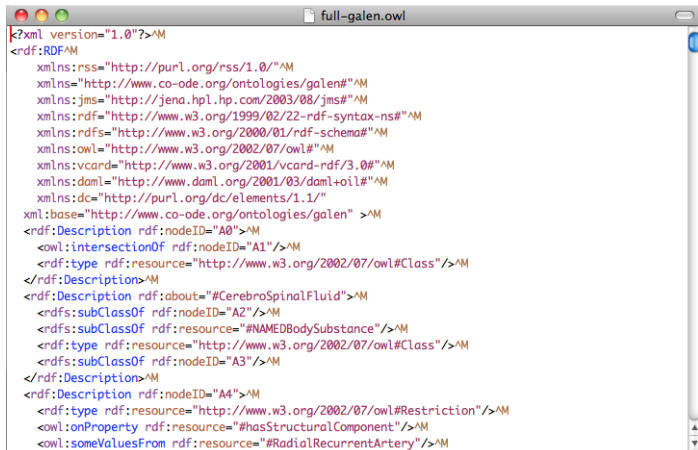


- EL+:
  - CEL (<http://lat.inf.tu-dresden.de/systems/cel>)
- ALB (ALC with inverse roles and Boolean operators on roles)
  - MSPASS (<http://www.cs.man.ac.uk/~schmidt/mypass/>)
- SHIQ:
  - KAON2 (<http://kaon2.semanticweb.org>)
- SROIQ:
  - FaCT++ (<http://owl.man.ac.uk/factplusplus/>)
  - Hermit (<http://hermit-reasoner.com>)
  - Pellet (<http://clarkparsia.com/pellet/>)
  - RacerPro (<http://racer-systems.com>)
- ...



# More tool support?

- development of ontologies
- editing an OWL ontology with RDF/XML syntax (full-galen.owl in a text editor)



```
full-galen.owl
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rss="http://purl.org/rss/1.0/"
  xmlns="http://www.co-ode.org/ontologies/galen#"
  xmlns:jms="http://jena.hpl.hp.com/2003/08/jms#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xml:base="http://www.co-ode.org/ontologies/galen" >
  <rdf:Description rdf:nodeID="A0">
    <owl:intersectionOf rdf:nodeID="A1"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
  <rdf:Description rdf:about="#CerebroSpinalFluid">
    <rdfs:subClassOf rdf:nodeID="A2"/>
    <rdfs:subClassOf rdf:resource="#NAMEDBodySubstance"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    <rdfs:subClassOf rdf:nodeID="A3"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A4">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Restriction"/>
    <owl:onProperty rdf:resource="#hasStructuralComponent"/>
    <owl:someValuesFrom rdf:resource="#RadialRecurrentArtery"/>
```





# Ontology Editor

- full-galen.owl in Protégé  
(<http://protege.stanford.edu>)

The screenshot shows the Protégé ontology editor interface. The title bar indicates the file is 'galen (http://www.co-ode.org/ontologies/galen) - [/Users/dirk/Documents/full-galen.owl]'. The main window is divided into several panes:

- Class hierarchy:** Shows a tree view of classes. The 'Complication' class is selected and highlighted in blue. Its parent is 'AbnormalPhenomenon', which is a subclass of 'Phenomenon'. Other classes visible include 'TopCategory', 'ApplicationCategory', 'ApplicationInformation', 'DomainCategory', 'ArbitrarilyConjoinedPhenomenon', 'ModifierConcept', 'InfantFeedingProblem', 'ActOutcome', 'Anonymous-403', 'AnthraxVaccine', 'AntidiureticAgent', 'AxillaryLymphadenopathy', 'BCGVaccine', 'Biopsy', and 'BodySystemPhenomenon'.
- Annotations:** Shows the annotations for the selected 'Complication' class. It is currently empty.
- Description:** Shows the logical description of the 'Complication' class. It is defined as:  
Equivalent classes:  
• AbnormalPhenomenon and (isComplicationOf some Phenomenon)  
Superclasses:  
Inherited anonymous classes:  
• Phenomenon and (hasAbnormalityStatus some abnormal)  
Members: (empty)

At the bottom of the window, there is a status bar that reads: 'No Reasoner set. Select a reasoner from the Reasoner menu' and a checked checkbox for 'Show Inferences'.



# Ontology Editor

- full-galen.owl in NeOn Toolkit (<http://neon-toolkit.org>)

The screenshot displays the NeOn Toolkit interface. On the left, the 'Ontology Navigator' shows a tree structure under 'NewOntologyProject [OWL2]' with 'galen' expanded to 'ApplicationAttribute', where 'hasCommonFinding' is selected. The main panel, 'Entity Properties', shows the configuration for 'hasCommonFinding' with the following details:

- URI:** `<http://www.co-ode.org/ontologies/galen#hasCommonFinding>`
- Domain:** The domain of this property is defined as the intersection of the entries below. Create new:
- Range:** The range of this property is defined as the intersection of the entries below. Create new:
- Characteristics:** Define if this property is functional, inverse functional, transitive or symmetric. Local:  Transitive Closure:

At the bottom, there are tabs for 'Domain and Range', 'Taxonomy', 'Annotations', and 'Source View'.



# And now ...

Part B (Thomas): overview on modularity in ontologies

