Introduction to Ontology Engineering

with emphasis on Semantic Web Technologies

C. Maria Keet

KRDB Research Center
Free University of Bozen-Bolzano, Italy

Masters Ontology Winter School 2010
KRR Group, Meraka Institute, South Africa, 26-30 July 2010

Housekeeping points

- This course consists of lectures and exercises
- Each lecture takes about 2.5 hours, labs 45 minutes
- Following the lectures will be easier when you have read the recommended reading beforehand and it is assumed the student is familiar with first order logic and conceptual data modelling, such as UML and ER
- The topics covered in this course are of an **introductory** nature and due to time constraints only a selection of core and elective topics will be addressed.
- These slides serve as a teaching aid, not as a neat summary
- Course webpage, with introduction, references, and schedule: `http://www.meteck.org/teaching/SA/MOWS10OntoEngCouse.html`

Outline

Part I

Introduction to ontologies

Outline

Background

– Aristotle and colleagues: **O**ntology
– Engineering: ontolog**ies** (count noun)

– Investigating reality, representing it
– Putting an engineering artifact to use

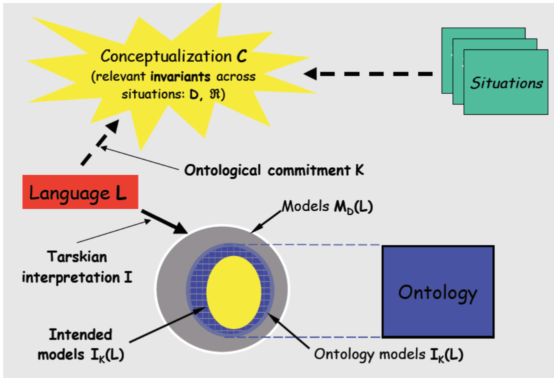
What then, is this engineering artifact?

(Guarino, 2002)

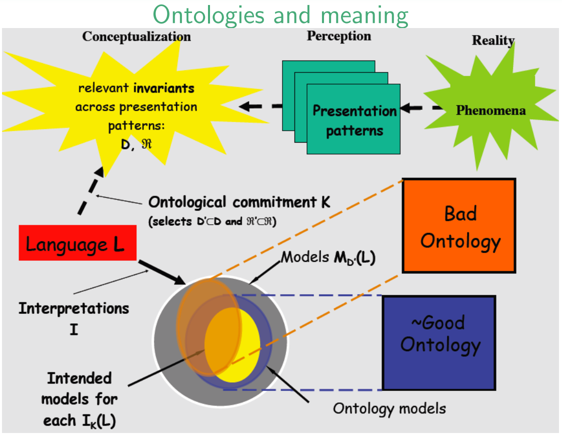
A few definitions

- Most quoted: "An ontology is a specification of a conceptualization" (by Tom Gruber, 1993)
- "a formal specification of a shared conceptualization" (by Borst, 1997)
- "An ontology is a formal, explicit specification of a shared conceptualization" (Studer et al., 1998)
- What is a *conceptualization*, and a *formal, explicit specification*? Why *shared*?

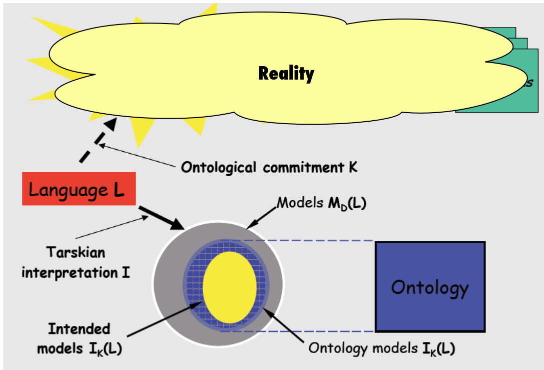
Ontologies and meaning



Ontologies and meaning



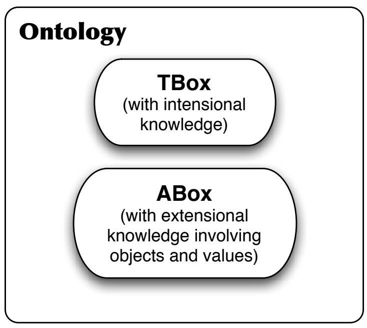
Ontologies and reality



More definitions

- More detailed: "An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary, i.e. its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models." (Guarino, 1998)
- And back to a simpler definition: "with an ontology being equivalent to a Description Logic knowledge base" (Horrocks et al, 2003)

Description Logic knowledge base



From logical to ontological level

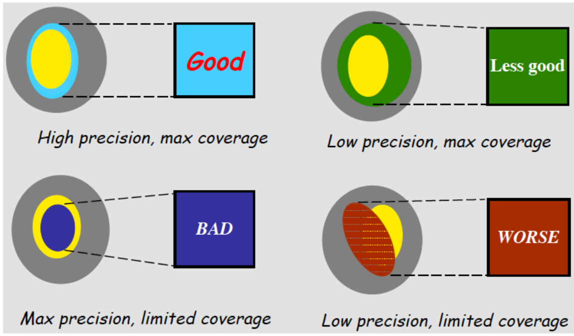
- Logical level (no structure, no constrained meaning[1]):
  - $\exists x(Apple(x) \land Red(x))$
- Epistemological level (structure, no constrained meaning):
  - $\exists x : apple\ Red(x)$ (many-sorted logics)
  - ~~$\exists x : red\ Apple(x)$~~
  - $Apple(a)$ and $hasColor(a, red)$ (description logics[2])
  - ~~$Red(a)$ and $hasShape(a, apple)$~~
- Ontological level (structure, constrained meaning):
  - Some structuring choices are excluded because of ontological constraints
  - *Apple* carries an identiy condition (and is a sortal), *Red* does not (is a qualia ['value'] of the quality ['attribute'] *hasColor* that a thing has)

adapted from (Guarino, 2008)

[1] well, meaning in the sense of subject domain semantics

[2] Likewise, DL has a formal, (model-theoretic) semantics, so the axioms have a meaning in that sense of 'meaning/semantics'

Quality of the ontology



(Guarino, 2002)

Quality of the ontology

- "Bad ontologies are (inter alia) those whose general terms lack the relation to corresponding universals in reality, and thereby also to corresponding instances." $\Rightarrow$ need for grounding
- "Good ontologies are reality representations, and the fact that such representations are possible is shown by the fact that, as is documented in our scientific textbooks, very many of them have already been achieved, though of course always only at some specific level of granularity and to some specific degree of precision, detail and completeness."

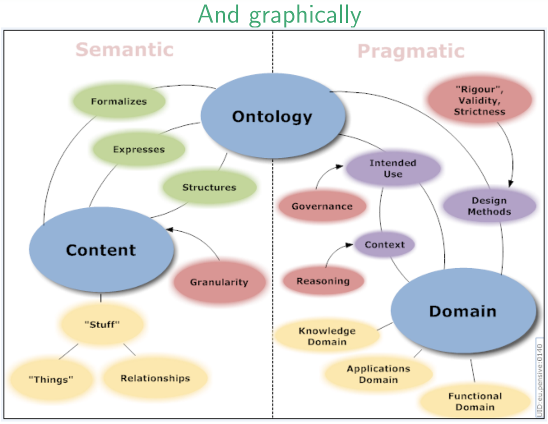
(Smith, 2004)

Initial Ontology Dimensions that have Evolved

- Semantic
  - Degree of Formality and Structure
  - Expressiveness of the Knowledge Representation Language
  - Representational Granularity
- Pragmatic
  - Intended Use
  - Role of Automated Reasoning
  - Descriptive vs. Prescriptive
  - Design Methodology
  - Governance

slide from, and more details available in: http://ontolog.cim3.net/file/work/OntologySummit2007/symposium/OntologyFramework_symposium--Gruninger-Obrst_20070424.ppt

And graphically



- Making, more or less precisely, the (dis-)agreement among people explicit
- Enrich software applications with the additional semantics
- Thus, practically, improving: computer-computer, computer-human, and human-human communication

Examples in different application areas, using different features

- **Data(base) integration** (example today)
- Instance classification (example today)
- Matchmaking and services
- Querying, information retrieval
  - Ontology-Based Data Access
  - Ontologies to improve NLP
- Bringing more quality criteria into conceptual data modelling to develop a better model (hence, a better quality software system)
- Orchestrating the components in semantic scientific workflows, e-learning, etc.

Success?

- Only if Berners-Lee's *vision* of the Semantic Web (as in the SciAm 2001 paper) has been realised?
  - How much "semantics" (with ontologies)?
  - SemWeb stack, technologies
- Absolute measures? e.g.,
  - Usage of Amazon's recommender system with and without ontologies
  - Information retrieval: compare precision and recall between a statistics-based and a ontologies-mediated document system
  - Feasibility and performance of a set of user queries posed to a RDBMS and its RDF-ised version
- Relative measures
  - According to whom is it a success?
    - philosopher, logician, engineer, domain expert, CEO...
  - What was taken as baseline material? e.g.,
    - from string search in a digital library to ontology-annotated sorting of query answer
    - from no or clustering-based instance classification to one with OWL-based knowledge bases

Early bioinformatics

- Advances in technologies to sequence genomes in the late '80s-early'90s, as well as more technologies for proteins
- Need to store the data: in databases ('90s)
- Several 'model organism' databases with genes (and genomes) of the fruitfly, yeast, mouse, a flowering plant, flatworm, zebrafish
- Compare genes and genomes
  - One observation (of many): About 12% (some 18,000) of the worm genes encode proteins whose biological roles could be inferred from their similarity to their (putative) orthologues in yeast, comprising about 27% of the yeast genes (about 5,700)
  - *What else can we infer from comparing genes and genomes (across species)?*
  - *What about the possibility of automated transfer of biological annotations from the model organisms to less 'fancy' organisms based on gene and protein sequence similarity, to use to improve human health or agriculture?*

Scope and requirements

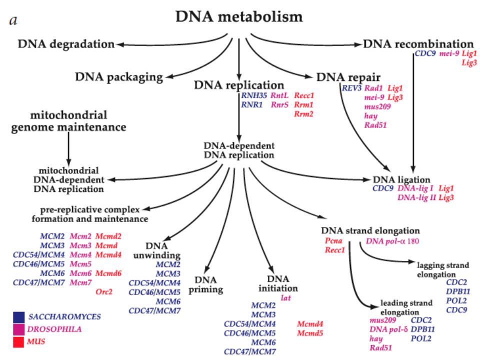
- Need: a mainly computational system for comparing or transferring annotation among different species
- Methods for sequence comparison existed
- Main requirements:
  - One needs a shared, controlled, vocabulary for annotation of the gene *products*, the *location* where they are active, the *function* they perform
  - To take on board and be compatible with existing terminologies, like gene and protein keyword databases such as UniProt, GenBank, Pfam, ENZYME etc.
  - Database interoperability among, at least, the model organism databases
  - Organize, describe, query and visualize biological knowledge at vastly different stages of completeness
  - Any system must be flexible and tolerant of this constantly changing level of knowledge and allow updates on a continuing basis

How to meet such requirements?

- Two main strands in activities:
  - Very early adopters from late 1990s (by sub-cellular bio), i.e., starting *without* Semantic Web Technologies
  - Early adopters from mid 2000s (e.g., eco and agri), starting *with* Semantic Web Technologies
- The Gene Ontology Consortium
  - Initiated by fly, yeast and mouse database curators[3] and others came on board (see `http://ww.geneontology.org` for a full list)
  - In the beginning, there was the flat file format .obo to store the ontologies, definitions of terms and gene associations
  - Several techniques on offer for data(base) integration that could be experimented with

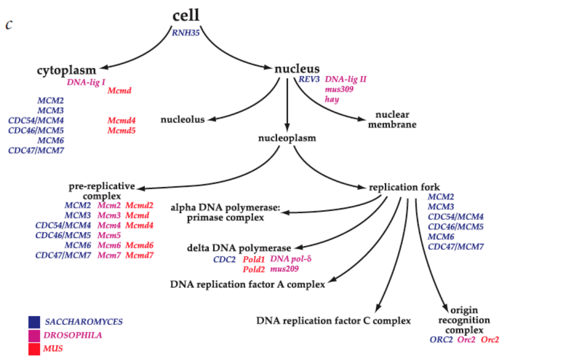
[3] more precisely: FlyBase (http://www.flybase.bio.indiana.edu), Berkeley Drosophila Genome Project (http://fruitfly.bdgp.berkeley.edu), Saccharomyces Genome Database (http://genome-www.stanford.edu), and Mouse Genome Database and Gene Expression Database (http://www.informatics.jax.org).

GO contents example (process)



from GOC, 2000

GO contents example (cellular component)



from GOC, 2000

Progress

- Tool development, e.g. to:
  - add and query its contents
  - annotate genes (semi-automatically)
  - link the three GO ontologies
  - mine the literature (NLP)
- Content development: more in the GO, extensions to the GO (e.g., rice traits), copy of the principle to other subject domains (e.g., zebrafish anatomy)
- The GO and its approach went well beyond the initial scope (which does not imply that all requirements were met fully)

Toward an update of the approach and contents

- Problems:
  - one can infer very little knowledge from the obo-based bio-ontologies (mainly where there are errors, but not *new* insights)—but note that that was not it's original aim
  - semantics of the relations overloaded
  - mushrooming of obo-based bio-ontologies by different communities, which makes interoperation of the ontologies difficult
  - greater needs for collaborative ontology development, maintenance, etc.
- Proposed solution: structured, coordinated, development of ontologies adhering to a set of principles: the OBO Foundry

OBO Foundry

- Extending the **O**pen **B**iological **O**ntologies principles...
  - open,
  - orthogonal,
  - same syntax,
  - common space for identifiers
- ... to one for the **O**pen **B**iological and Biomedical **O**ntologies:
  - developed in a collaborative effort
  - usage of common relations that are unambiguously defined (*in casu*: the Relation Ontology)
  - provide procedures for user feedback and for identifying successive versions
  - has to have a clearly bounded subject-matter ("so that an ontology devoted to cell components, for example, should not include terms like 'database' or 'integer' " ...)

More info in Smith et al, 2007, and `http://www.obofoundry.org`

OBO Foundry coverage (canonical ontologies)

RELATION TO TIME / GRANULARITY	CONTINUANT				OCCURRENT
	INDEPENDENT		DEPENDENT		
ORGAN AND ORGANISM	Organism (NCBI Taxonomy?)	Anatomical Entity (FMA, CARO)	Organ Function (FMP, CPRO)	Phenotypic Quality (PaTO)	Organism-Level Process (GO)
CELL AND CELLULAR COMPONENT	Cell (CL)	Cellular Component (FMA, GO)	Cellular Function (GO)		Cellular Process (GO)
MOLECULE	Molecule (ChEBI, SO, RnaO, PrO)		Molecular Function (GO)		Molecular Process (GO)

OBO Foundry

- Sorting out the ontologies we have; e.g.,
  - harmonizing the four cell type ontologies into one (CL)
  - coordinating the anatomy ontologies of the various model organisms through a Common Aanatomy Reference Ontology
  - modularization of the subject domain by granularity, continuants, and occurents
- Adding ontologies to fill the gaps
- making OBO and OWL ontologies interoperable
- "Our long-term goal is that the data generated through biomedical research should form a single, consistent, cumulatively expanding and algorithmically tractable whole"
- "The result is an expanding family of ontologies designed to be interoperable and logically well formed and to incorporate accurate representations of biological reality"
- Aimed at "coordinated evolution of ontologies to support biomedical data integration"

Instance classification with protein phosphatases (Wolstencroft et al, 2007)

- The setting:
  - Lots of sequence data in data silos that needs to be enriched with biological knowledge
  - Need to organise and classify genes and proteins into functional groups to compare typical properties across species
- The problems:
  - There is no proper, real life, use case that demonstrates the benefits of DL reasoning services such as taxonomic and instance classification
  - Limitations of traditional similarity methods, and automated protein motif and domain matching
  - Automation of p-domain analysis, but not for its interpretation (i..e, detects presence but not consequences for sub-family membership)

Idea

- Maybe OWL reasoning can help with the interpretation of the analysis results:
  - That it does the classification of the (family of) proteins as good as a human expert for organisms x (in casu, human)
  - That the approach is 'transportable' to classification of the (family of) proteins in another organism of which much less is known (in casu, *Aspergillus fumigatus*), hence make predictions for those instances by means of classifying them
- Use taxonomic classification and instance classification reasoning services

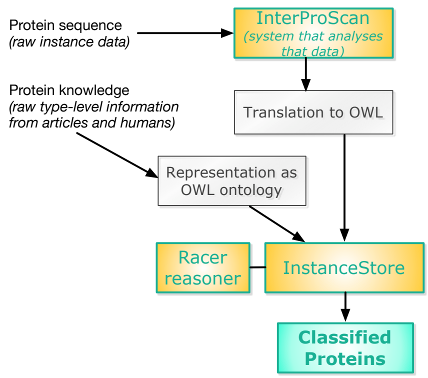
How it can be done

- Develop ontology for the subject domain, in OWL
  - Extract knowledge from peer-reviewed literature
  - Protein phosphatases; e.g.

```
Class R5Phosphatase Complete
  (Protein and
    (hasDomain two TyrosinePhosphataseCatalyticDomain) and
    (hasDomain some TransmembraneDomain) and
    (hasDomain some FibronectinDomain) and
     (hasDomain some CarbonicAnhydraseDomain) and
     hasDomain only (TyrosinePhosphataseCatalyticDomain and
      TransmembraneDomain and
      CarbonicAnhydraseDomain))
```

- Obtain instance data
  - Process protein sequences by InterProScan
  - Transform into OWL
- Put it together in some system with a reasoner
  - InstanceStore
  - Racer reasoner

Sequence of activities and architecture



Results

- Human phosphatases:
  - The reasoner as good as human expert classification
  - Identification of additional p-domains, refined the classification into further subtypes
- *A. fumigatus* phosphatates:
  - Some phosphatases did not fit in any class, representing differences between the human and *A. fumigatus* protein families
  - Identification of a novel type of calcineurin phosphatase (has extra domain, like in other pathogenic fungi)
- Overall: demonstration that ontology-based approach with automated reasoning has some advantages over (in addition to the) existing technologies & human labour, and resulted in discovery of novel biological information

Part II

Ontology Languages: OWL and OWL2

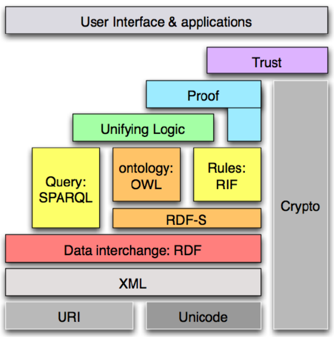
Outline

4. Introduction
   - Limitations of RDFS
5. OWL
   - Design of OWL
   - OWL and Description Logics
   - OWL Syntaxes
6. Limitations
7. OWL 2
   - OWL 2 DL
8. OWL 2 profiles
   - OWL 2 EL
   - OWL 2 QL
   - OWL 2 RL
9. Reasoning

Toward one ontology language

- Plethora of ontology languages; e.g., KIF, KL-ONE, LOOM, F-logic, DAML, OIL, DAML+OIL, ...
- Lack of a lingua franca; hence, ontology interoperation problems even on the syntactic level
- Advances in expressive DL languages and, more importantly, in automated reasoners for expressive DL languages (mainly: FaCT++, then Racer)
- Limitations of RDF(S) as Semantic Web 'ontology language'

The place of RFDS in the layer cake



RDFS as an Ontology Language

- Classes
- Properties
- Class hierarchies
- Property hierarchies
- Domain and range restrictions

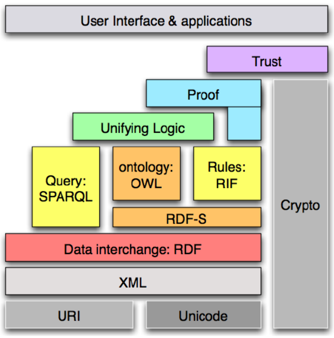
Expressive limitations of RDF(S)

- Only binary relations
- Characteristics of Properties (e.g. inverse, transitive, symmetric)
- Local range restrictions (e.g. for Class Person, the property hasName has range xsd:string)
- Complex concept descriptions (e.g. Person is defined by Man and Woman)
- Cardinality restrictions (e.g. a Person may have at most 1 name)
- Disjointness axioms (e.g. nobody can be both a Man and a Woman)

Layering issues

- Syntax
  - Only binary relations in RDF
  - Verbose Syntax
  - No limitations on graph in RDF
    - Every graph is valid
- Semantics
  - Malformed graphs
  - Use of vocabulary in language
    - e.g. ⟨rdfs:Class,rdfs:subClassOf,ex:a⟩
  - Meta-classes
    - e.g. ⟨ex:a,rdf:type,ex:a⟩

The place of OWL in the layer cake



Stack of Languages

- XML
  - Surface syntax, no semantics
- XML Schema
  - Describes structure of XML documents
- RDF
  - Datamodel for "relations" between "things"
- RDF Schema
  - RDF Vocabulary Definition Language
- OWL
  - A more expressive Vocabulary Definition Language

Design Goals for OWL

- **Shareable**
- **Changing** over time
- **Interoperability**
- **Inconsistency** detection
- Balancing **expressivity and complexity**
- **Ease of use**
- Compatible with **existing standards**
- **Internationalization**

Requirements for OWL

- Ontologies are **object on the Web**
- with **their own meta-data**, versioning, etc...
- Ontologies are **extendable**
- They contain **classes, properties, data-types, range/domain, individuals**
- **Equality** (for classes, for individuals)
- **Classes as instances**
- **Cardinality** constraints
- **XML** syntax

Objectives for OWL

Objectives:

- layered language
- complex datatypes
- digital signatures
- decidability (in part)
- local unique names (in part)

Disregarded:

- default values
- closed world option
- property chaining
- arithmetic
- string operations
- partial imports
- view definitions
- procedural attachments

Extending RDF Schema

- Leveraging experiences with OWL's predecessors SHOE, OIL, DAML-ONT, and DAML+OIL (frames, OO, DL)
- OWL extends RDF Schema to a full-fledged knowledge representation language for the Web
  - Logical expressions (and, or, not)
  - (in)equality
  - local properties
  - required/optional properties
  - required values
  - enumerated classes
  - symmetry, inverse

Species of OWL

- OWL Lite
  - Classification hierarchy
  - Simple constraints
- OWL DL
  - Maximal expressiveness
  - While maintaining tractability
  - Standard formalization in a DL
- OWL Full
  - Very high expressiveness
  - Losing tractability
  - All syntactic freedom of RDF (self-modifying)

Features of OWL languages

- OWL Lite
  - (sub)classes, individuals
  - (sub)properties, domain, range
  - conjunction
  - (in)equality
  - (unqualified) cardinality 0/1
  - datatypes
  - inverse, transitive, symmetric properties
  - someValuesFrom
  - allValuesFrom
- OWL DL
  - Negation
  - Disjunction
  - (unqualified) Full cardinality
  - Enumerated classes
  - hasValue
- OWL Full
  - Meta-classes
  - Modify language

OWL Full

- **No restriction on use of vocabulary** (as long as legal RDF)
  - Classes as instances (and much more)
- **RDF style model theory**
  - Reasoning using FOL engine
  - Semantics should correspond to OWL DL for restricted KBs

OWL DL

- Use of vocabulary restricted
  - Cannot be used to do "nasty things" (e.g., modify OWL)
  - No classes as instances (this will be discussed in a later lecture)
  - Defined by abstract syntax
- Standard DL-based model theory
  - Direct correspondence with a DL
  - Automated reasoning with DL reasoners (e.g., Racer, Pellet, FaCT^{++})

OWL Lite

- No explicit negation or union
- Restricted cardinality (0/1)
- No nominals (oneOf)
- DL-based semantics
  - Automated reasoning with DL reasoners (e.g., Racer, Pellet, FaCT^{++})

More on OWL species

- OWL Full is *not* a Description Logic
- OWL Lite has strong syntactic restrictions, but only limited semantics restrictions cf. OWL DL
  - Negation can be encoded using disjointness
  - With negation an conjunction, you can encode disjunction
- For instance:

```
Class(C complete unionOf(B C))
```

is equivalent to:

```
DisjointClasses(notB B)
DisjointClasses(notC C)
Class(notBandnotC complete notB notC)
DisjointClasses(notBandnotC BorC)
Class(C complete notBandnotC)
```

More on layering and OWL flavours

- For an OWL DL-restricted KB, OWL Full semantics is **not** equivalent to OWL DL semantics

John friend Susan .

OWL Full entails:

John rdf:type owl:Thing . Susan rdf:type owl:Thing . friend rdf:type owl:ObjectProperty .

John rdf:type _:x . _:x owl:onProperty friend . _:x owl:minCardinality "1"^^xsd:nonNegativeInteger .

OWL and Description Logics

- OWL Lite corresponds to the DL $\mathcal{SHIF}(\mathbf{D})$
  - Named classes (A)
  - Named properties (P)
  - Individuals ($C(o)$)
  - Property values ($P(o, a)$)
  - Intersection ($C \sqcap D$)
  - Union ($C \sqcup D$)
  - Negation ($\neg C$)
  - Existential value restrictions ($\exists P.C$)
  - Universal value restrictions ($\forall P.C$)
  - Unqualified (0/1) number restrictions ($\geq nP$, $\leq nP$, $= nP$), $0 \leq n \leq 1$
- OWL DL corresponds to the DL $\mathcal{SHOIN}(\mathbf{D})$
  - Arbitrary number restrictions ($\geq nP$, $\leq nP$, $= nP$), $0 \leq n$
  - Property value ($\exists P.\{o\}$)
  - Enumeration ($\{o_1, ..., o_n\}$)

OWL constructs

OWL Construct	DL	Example
intersectionOf	$C_1 \sqcap ... \sqcap C_n$	$Human \sqcap Male$
unionOf	$C_1 \sqcup ... \sqcup C_n$	$Doctor \sqcup Lawyer$
complementOf	$\neg C$	$\neg Male$
oneOf	$\{o_1, ..., o_n\}$	$\{john, mary\}$
allValuesFrom	$\forall P.C$	$\forall hasChild.Doctor$
someValuesFrom	$\exists P.C$	$\exists hasChild.Lawyer$
value	$\exists P.\{o\}$	$\exists citizenOf.USA$
minCardinality	$\geq nP.C$	$\geq 2hasChild.Lawyer$
maxCardinality	$\leq nP.C$	$\leq 1hasChild.Male$
cardinality	$= nP.C$	$= 1hasParent.Female$

+ XML Schema datatypes: int, string, real, etc...

OWL axioms

OWL Axiom	DL	Example
SubClassOf	$C_1 \sqsubseteq C_2$	$Human \sqsubseteq Animal \sqcap Biped$
EquivalentClasses	$C_1 \equiv ... \equiv C_n$	$Man \equiv Human \sqcap Male$
SubPropertyOf	$P_1 \sqsubseteq P_2$	$hasDaughter \sqsubseteq hasChild$
EquivalentProperties	$P_1 \equiv ... \equiv P_n$	$cost \equiv price$
SameIndividual	$o_1 = ... = o_n$	$President\_Bush = G\_W\_Bush$
DisjointClasses	$C_i \sqsubseteq \neg C_j$	$Male \sqsubseteq \neg Female$
DifferentIndividuals	$o_i \neq o_j$	$john \neq peter$
inverseOf	$P_1 \equiv P_2^-$	$hasChild \equiv hasParent^-$
Transitive	$P^+ \sqsubseteq P$	$ancestor^+ \sqsubseteq ancestor$
Symmetric	$P \equiv P^-$	$connectedTo \equiv connectedTo^-$

DL-based OWL species as Semantic Web languages vs DLs

⇒ OWL uses URI references as names (like used in RDF, e.g., `http://www.w3.org/2002/07/owl#Thing`)

⇒ OWL gathers information into ontologies stored as documents written in RDF/XML, things like `owl:imports`

⇒ RDF data types and XML schema data types for the ranges of data properties (attributes) (`DataPropertyRange`)

- OWL-DL and OWL-Lite with a frame-like abstract syntax, whereas RDF/XML is the official exchange syntax for OWL
- Annotations

Syntaxes of OWL

- RDF
  - Official exchange syntax
  - Hard for humans
  - RDF parsers are hard to write!
- XML
  - Not the RDF syntax
  - Still hard for humans, but more XML than RDF tools available
- Abstract syntax
  - Not defined for OWL Full
  - To some, considered human readable
- User-usable ones
  - e.g., Manchester syntax, informal and limited matching with UML

OWL in RDF/XML

Example from [OwlGuide]:

<!ENTITY vin "http://www.w3.org/TR/2004/REC−owl−guide−20040210/wine#" >
<!ENTITY food "http://www.w3.org/TR/2004/REC−owl−guide−20040210/food#" > ...
<rdf:RDF
xmlns:vin="http://www.w3.org/TR/2004/REC−owl−guide−20040210/wine#"
xmlns:food="http://www.w3.org/TR/2004/REC−owl−guide−20040210/food#"
... >

<owl:Class rdf:ID="Wine"> <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/> <rdfs:label xml:lang="en">wine</rdfs:label> <rdfs:label xml:lang="fr">vin</rdfs:label> ... </owl:Class>

<owl:Class rdf:ID="Pasta"> <rdfs:subClassOf rdf:resource="#EdibleThing" /> ... </owl:Class> </rdf:RDF>

OWL Abstract syntax

Class(professor partial) Class(associateProfessor partial academicStaffMember)

DisjointClasses (associateProfessor assistantProfessor)
DisjointClasses (professor associateProfessor)

Class(faculty complete academicStaffMember)

In DL syntax:

associateProfessor $\sqsubseteq$ academicStaffMember
associateProfessor $\sqsubseteq \neg$ assistantProfessor
professor $\sqsubseteq \neg$ associateProfessor

faculty $\equiv$ academicStaffMember

More examples

DatatypeProperty(age range(xsd:nonNegativeInteger))
ObjectProperty(lecturesIn)

ObjectProperty(isTaughtBy domain(course) range(academicStaffMember))
SubPropertyOf(isTaughtBy involves)

ObjectProperty(teaches inverseOf(isTaughtBy) domain(academicStaffMember) range(course))

EquivalentProperties (lecturesIn teaches)

ObjectProperty(hasSameGradeAs Transitive Symmetric domain(student) range(student))

More examples

In DL syntax:

$\top \sqsubseteq \forall age.xsd : nonNegativeInteger$
$\top \sqsubseteq \forall isTaughtBy^{-}.course$
$\top \sqsubseteq \forall isTaughtBy.academicStaffMember$
$isTaughtBy \sqsubseteq involves$
$teaches \equiv isTaughtBy^{-}$
$\top \sqsubseteq \forall teaches^{-}.academicStaffMember$
$\top \sqsubseteq \forall teaches.course$
$lecturesIn \equiv teaches$
$hasSameGradeAs^{+} \sqsubseteq hasSameGradeAs$
$hasSameGradeAs \equiv hasSameGradeAs^{-}$
$\top \sqsubseteq \forall hasSameGradeAs^{-}.student$
$\top \sqsubseteq \forall hasSameGradeAs.student$

More examples

Individual (949318 type(lecturer))

Individual (949352 type(academicStaffMember) value(age "39"^^&xsd;integer))

ObjectProperty(isTaughtBy Functional)

Individual (CIT1111 type(course) value(isTaughtBy 949352) value(isTaughtBy 949318))

DifferentIndividuals (949318 949352) DifferentIndividuals (949352 949111 949318)

More examples

In DL syntax:

$949318 : lecturer$
$949352 : academicStaffMember$
$\langle 949352, "39"\hat{}\hat{}\&xsd; integer\rangle : age$
$\top \sqsubseteq \leq 1 isTaughtBy$
$CIT1111 : course$
$\langle CIT1111, 949352\rangle : isTaughtBy$
$\langle CIT1111, 949318\rangle : isTaughtBy$
$949318 \neq 949352$
$949352 \neq 949111$
$949111 \neq 949318$
$949352 \neq 949318$

More examples

Class(firstYearCourse partial restriction (isTaughtBy allValuesFrom (Professor)))

Class(mathCourse partial restriction (isTaughtBy hasValue (949352)))

Class(academicStaffMember partial restriction (teaches someValuesFrom (undergraduateCourse)))

Class(course partial restriction (isTaughtBy minCardinality (1)))

Class(department partial restriction (hasMember minCardinality(10)) restriction (hasMember maxCardinality(30)))

More examples

In DL syntax:

$firstYearCourse \sqsubseteq \forall isTaughtBy.Professor$
$mathCourse \sqsubseteq \exists isTaughtBy.\{949352\}$
$academicStaffMember \sqsubseteq \exists teaches.undergraduateCourse$
$course \sqsubseteq \geq 1 isTaughtBy$
$department \sqsubseteq \geq 10 hasMember \sqcap \leq 30 hasMember$

More examples

Class(course partial complementOf(staffMember))

Class(peopleAtUni complete unionOf(staffMember student))

Class(facultyInCS complete intersectionOf(faculty restriction (belongsTo hasValue (CSDepartment))))

Class(adminStaff complete intersectionOf(staffMember complementOf(unionOf(faculty techSupportStaff))))

In DL syntax:

$course \sqsubseteq \neg staffMember$
$peopleAtUni \equiv staffMember \sqcup student$
$facultyInCS \equiv faculty \sqcap \exists belongsTo.\{CSDepartment\}$
$adminStaff \equiv staffMember \sqcap \neg(faculty \sqcup techSupportStaff)$

Layering on top of RDF(S)

- RDF(S) bottom layer in Semantic Web stack
- Higher languages *layer* on top of RDFS

Syntactic Layering

- *Every valid RDF statement is a valid statement in a higher language*
- This *includes* triples containing keywords of these languages(!)

Semantic Layering

For RDFS graph G and higher-level language L:
If $G \models_{RDFS} G'$ then $G \models_L G'$, and *ideally*
if $G \models_L G'$ then $G \models_{RDFS} G'$

Syntactically layering OWL on RDF(S)

OWL Lite, OWL DL

- OWL Lite, OWL DL subsets of RDF
- Allowed triples defined through mapping from abstract syntax
- *Partial* layering:
  - *every* OWL Lite/DL ontology is an RDF graph
  - *some* RDF graphs are OWL Lite/DL ontologies

OWL Full

- OWL Full encompasses RDF
- *Complete* layering:
  - *every* OWL Full is an RDF graph
  - *all* RDF graphs are OWL Full ontologies

Semantically layering OWL on RDF(S)

OWL Lite, OWL DL

- OWL Lite/DL semantics *not* related to RDFS semantics
- Redefine semantics of RDFS keywords, e.g., `rdfs:subClassOf`
- Work ongoing to describe correspondence between subset of RDFS and OWL Lite/DL

OWL Full

- OWL Full semantics is *extension* of RDFS semantics
- OWL Full is undecidable
- OWL Full semantics hard to understand

OWL Lite/DL vs. RDF

- RDF Graph defined through translation from Abstract Syntax
- Example:

Class(Human partial Animal
restriction(hasLegs cardinality(2))
restriction(hasName allValuesFrom(xsd:string)))

Human	rdf:type	owl:Class
Human	rdfs:subClassOf	Animal
Human	rdfs:subClassOf	_:X1
_:X1	rdf:type	owl:Restriction
_:X1	owl:onProperty	hasLegs
_:X1	owl:cardinality	"2"8sd:nonNegativeInteger
Human	rdfs:subClassOf	_:X2
_:X2	rdf:type	owl:Restriction
_:X2	owl:onProperty	hasName
_:X2	owl:allValuesFrom	xsd:string

OWL Lite/DL vs. RDF

- Not every RDF graph is OWL Lite/DL ontology
- Example:

A rdf:type A

- How to check whether an RDF graph G is OWL DL?
  1. Construct an OWL ontology O in Abstract Syntax
  2. Translate to RDF graph G'
  3. If G=G', then G is OWL DL
     - Otherwise, go to step (1)

Expressivity limitations

- Qualified cardinality restrictions (e.g., no `Bicycle` $\sqsubseteq\ \geq 2$ `hasComponent.Wheel`)
- Relational properties (no reflexivity, irreflexivity)
- Data types
  - restrictions to a subset of datatype values (ranges)
  - relationships between values of data properties on one object
  - relationships between values of data properties on different objects
  - aggregation functions
- Other things like annotations, imports, versioning, species validation (see p315 of the paper)

Syntax problems

- Having both frame-based legacy (Abstract syntax) and axioms (DL) was deemed confusing
- Type of ontology entity. e.g.,

```
Class(A partial
    restriction(hasB someValuesFrom(C))
```

  - `hasB` is data property and `C` a datatype?
  - `hasB` an object property and `C` a class?

  OWL-DL has a strict separation of the vocabulary, but the specification does not precisely specify how to enforce this separation at the syntactic level

More syntax problems

- RDF's triple notation, difficult to read and process
- OWL 1 provides mapping from the Abstract Syntax into OWL RDF, but not the converse:
  - an RDF graph G is an OWL-DL ontology if there exists an ontology $\mathcal{O}$ in Abstract Syntax s.t. the result of the normative transformation of $\mathcal{O}$ into triples is precisely G, which makes checking whether G is an OWL-DL ontology very hard in practice:
  - examine all 'relevant' ontologies $\mathcal{O}$ in abstract syntax, check whether the normative transformation of $\mathcal{O}$ into RDF yields precisely G.

Problems with the semantics

- RDF's blank nodes, but unnamed individuals not directly available in $\mathcal{SHOIN}(\mathcal{D})$
- Frames and axioms

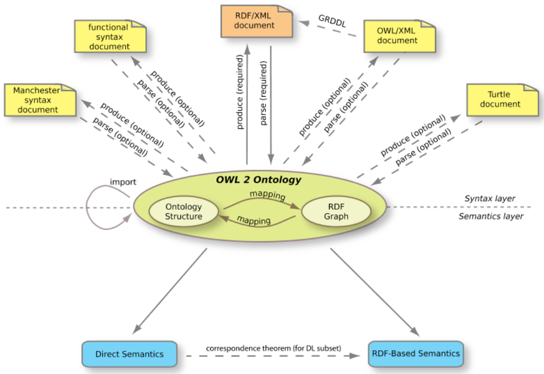
Aims

- Address as much as possible of the identified problems (previous slides and "the next steps for OWL 2" paper)
- Task: compare this with the possible "future extensions" of the "the making of an ontology language" paper

Some general points

- OWL 2 a W3C recommendation since 27-10-'09
- Any OWL 2 ontology can also be viewed as an RDF graph (The relationship between these two views is specified by the Mapping to RDF Graphs document)
- Direct, i.e. model-theoretic, semantics (⇒ OWL 2 DL) and an RDF-based semantics (⇒ OWL 2 full)
- Primary exchange syntax for OWL 2 is RDF/XML, others are optional
- Three profiles, which are sub-languages of OWL 2 (syntactic restrictions)

The Structure of OWL 2



Overview

- Based on $\mathcal{SROIQ}(D)$, which is 2NExpTime-complete
- More expressive than OWL-DL
- Fancier metamodelling and annotations
- Improved ontology publishing, imports and versioning control
- Variety of syntaxes, RDF serialization (but no RDF-style semantics)

The language: properties of properties

- property chains (`ObjectPropertyChain`), e.g.:
  ```
  SubObjectPropertyOf( ObjectPropertyChain(
     a:hasMother a:hasSister ) a:hasAunt )
  ```
  with having Lois as the mother of Stewie, and Carol a sister of Lois, the ontology entails that Stewie has Carol as aunt
- ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, ObjectHasSelf, FunctionalObjectProperty, InverseFunctionalObjectProperty, IrreflexiveObjectProperty, AsymmetricObjectProperty, and DisjointObjectProperties **only on simple object properties** (i.e., has no direct or indirect subproperties that are either transitive or are defined by means of property chains—so we still can't represent parthood fully)

The language: other extensions

- qualified cardinality restrictions
- The `Haskey` 'key' that are **not** keys like in conceptual models and databases
  - Alike inverse functional only (i.e., merely 1:n instead of 1:1) but applicable only to individuals that are explicitly named in an ontology
  - No unique name assumption, hence inferences are different from that expected of keys in databases
  - "relevant mainly for query answering" [Cuenca Grau et al, 2008, p316], which does not go well with OWL 2 DL in non-toy applications anyway
- Richer datatypes, data ranges; e.g., `DatatypeRestriction( xsd:integer xsd:minInclusive "5"8sd:integer xsd:maxExclusive "10"8sd:integer )`

Partial table of features

Language ⇒ **Feature** ⇓	OWL 1 **Lite**	OWL 1 **DL**	OWL 2 **DL**	OWL 2 Profiles EL	OWL 2 Profiles QL	OWL 2 Profiles RL
Role hierarchy	+	+	+	.	+	.
N-ary roles (where $n \geq 2$)	−	−	−	.	?	.
Role chaining	−	−	+	.	−	.
Role acyclicity	−	−	−	.	−	.
Symmetry	+	+	+	.	+	.
Role values	−	−	−	.	−	.
Qualified number restrictions	−	−	+	.	−	.
One-of, enumerated classes	?	+	+	.	−	.
Functional dependency	+	+	+	.	?	.
Covering constraint over concepts	?	+	+	.	−	.
Complement of concepts	?	+	+	.	+	.
Complement of roles	−	−	+	.	+	.
Concept identification	−	−	−	.	−	.
Range typing	−	+	+	.	+	.
Reflexivity	−	−	+	.	−	.
Antisymmetry	−	−	−	.	−	.
Transitivity	+	+	+	.	−	.
Asymmetry	?	?	+	−	+	+
Irreflexivity	−	−	+	.	−	.
.	.	.	.	.	.	.

Exercise: verify the question marks in the table (tentatively all "–") and fill in the dots (any "±" should be qualified at to what the restriction is)

Rationale

- Computational considerations
  - Consult "OWL profiles" page *Table 10. Complexity of the Profiles*
- Robustness of implementations w.r.t. *scalable* applications
- Already enjoy 'substantial' user base

OWL 2 EL Overview

- Intended for large 'simple' ontologies
- Focussed on type-level knowledge (TBox)
- Better computational behaviour than OWL 2 DL (polynomial vs. exponential/open)
- Based on the DL language $\mathcal{EL}^{++}$ (PTime complete)
- Reasoner: e.g. CEL `http://code.google.com/p/cel/`

Supported class restrictions

- existential quantification to a class expression or a data range
- existential quantification to an individual or a literal
- self-restriction
- enumerations involving a single individual or a single literal
- intersection of classes and data ranges

Supported axioms, restricted to allowed set of class expressions

- class inclusion, equivalence, disjointness
- object property inclusion (w. or w.o. property chains), and data property inclusion
- property equivalence
- transitive object properties
- reflexive object properties
- domain and range restrictions
- assertions
- functional data properties
- keys

NOT supported in OWL 2 EL

- universal quantification to a class expression or a data range
- cardinality restrictions
- disjunction
- class negation
- enumerations involving more than one individual
- disjoint properties
- irreflexive, symmetric, and asymmetric object properties
- inverse object properties, functional and inverse-functional object properties

OWL 2 QL Overview

- Query answering over a large amount of instances with same kind of performance as relational databases (Ontology-Based Data Access)
- Expressive features cover several used features of UML Class diagrams and ER models (‘COnceptual MOdel-based Data Access’)
- Based on *DL-Lite*$_{\mathcal{R}}$ (*more is possible with UNA and in some implementations*)

Supported Axioms in OWL 2QL, restrictions

- Subclass expressions restrictions:
  - a class
  - existential quantification (ObjectSomeValuesFrom) where the class is limited to owl:Thing
  - existential quantification to a data range (DataSomeValuesFrom)
- Super expressions restrictions:
  - a class
  - intersection (ObjectIntersectionOf)
  - negation (ObjectComplementOf)
  - existential quantification to a class (ObjectSomeValuesFrom)
  - existential quantification to a data range (DataSomeValuesFrom)

Supported Axioms in OWL 2QL

- Restrictions on class expressions, object and data properties occurring in functionality assertions cannot be specialized
- subclass axioms
- class expression equivalence (involving subClassExpression), disjointness
- inverse object properties
- property inclusion (not involving property chains and SubDataPropertyOf)
- property equivalence
- property domain and range
- disjoint properties
- symmetric, reflexive, irreflexive, asymmetric properties
- assertions other than individual equality assertions and negative property assertions (DifferentIndividuals, ClassAssertion, ObjectPropertyAssertion, and DataPropertyAssertion)

NOT supported in OWL 2 QL

- existential quantification to a class expression or a data range in the subclass position
- self-restriction
- existential quantification to an individual or a literal
- enumeration of individuals and literals
- universal quantification to a class expression or a data range
- cardinality restrictions
- disjunction
- property inclusions involving property chains
- functional and inverse-functional properties
- transitive properties
- keys
- individual equality assertions and negative property assertions

OWL 2 RL Overview

- Development motivated by: what fraction of OWL 2 DL can be expressed by rules (with equality)?
- Scalable reasoning in the context of RDF(S) application
- Rule-based technologies (forward chaining rule system, over *instances*)
- Inspired by Description Logic Programs and pD*
- Reasoning in PTime

Supported in OWL 2 RL

- More restrictions on class expressions (see table 2, e.g. no SomeValuesFrom on the right-hand side of a subclass axiom)
- All axioms in OWL 2 RL are constrained in a way that is compliant with the restrictions in Table 2.
- Thus, OWL 2 RL supports all axioms of OWL 2 apart from disjoint unions of classes and reflexive object property axioms.
- No $\forall$ and $\neg$ on lhs, and $\exists$ and $\sqcup$ on rhs of $\sqsubseteq$

Another section on speculation about future extensions

- The 'leftover' from OWL 1's "Future extensions" (UNA, CWA, defaults), parthood relation (primarily: antisymmetry, restrictions on current usage of properties)
- New "future of OWL", a.o.:
  - Syntactic sugar: 'macros', 'n-aries'
  - Query languages: EQL-lite and nRQL w.r.t. SPARQL
  - Integration with rules: RIF, DL-safe rules, SBVR
  - Orthogonal dimensions: temporal, fuzzy, rough, probabilistic

Reasoning services for DL-based OWL ontologies

- OWL ontology is a first-order logical theory ⇒ verifying the formal properties of the ontology corresponds to reasoning over a first-order theory
- Main ('standard') reasoning tasks for the OWL ontologies:
  - consistency of the ontology
  - concept (and role) consistency
  - concept (and role) subsumption
  - instance checking
  - instance retrieval
  - query answering
- Non-standard reasoning services, such as explanation, repair, least common subsumer, ...
- Note: Not all OWL languages are equally suitable for all these reasoning tasks

Reasoning services for DL-based OWL ontologies

- Consistency of the ontology
  - Is the ontology $K = (T, A)$ consistent (non-selfcontradictory), i.e., is there at least a model for K?
- Concept (and role) consistency
  - is there a model of T in which C (resp. R) has a nonempty extension?
- Concept (and role) subsumption
  - i.e., is the extension of C (resp. R) contained in the extension of D in every model of T?
- Instance checking
  - is a a member of concept C in K, i.e., is the fact $C(a)$ satisfied by every interpretation of K?
- Instance retrieval
  - find all members of C in K, i.e., compute all individuals a s.t. $C(a)$ is satisfied by every interpretation of K
- Query answering
  - compute all tuples of individuals t s.t. query $q(t)$ is entailed by K, i.e., $q(t)$ is satisfied by every interpretation of K

Note: Reasoning with OWA (vs. CWA)

- **O**pen **W**orld **A**ssumption
  - Absence of information is interpreted as unknown information
  - Assumes incomplete information
  - Good for describing knowledge in a way that is extensible
- **C**losed **W**orld **A**ssumption
  - Absence of information is interpreted as negative information
  - Assumes we have complete information
  - Good for constraining information and validating data in an application

Example

Which alumni do not have a PhD?

Alumnus	Degree Obtained
Delani	PhD in history
Maria	PhD in politics
Peter	MSc in Informatics
Dalila	PhD in politics

- Query under CWA says "Peter"
- Query under OWA cannot say "Peter", because we do not know if Peter also obtained a PhD. To retrieve "Peter" we have add an axiom somehow stating that Peter does not have a PhD (e.g., by being an instance of *PhD student*, declaring the degrees to be disjoint & covering, ...).

Summary

Part III

Foundational and top-down aspects of ontology engineering

Outline

General notion

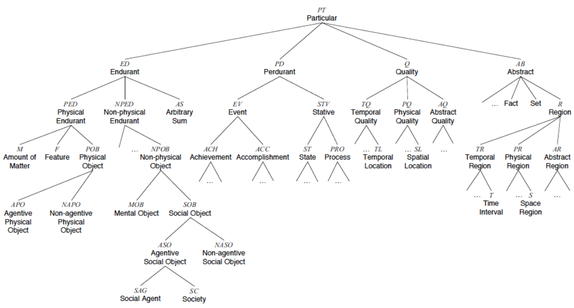
- Provide a top-level with basic categories of kinds of things
- Principal choices
  - Endurantist vs. Perdurantist
  - Universals vs. Particulars
- Formal...
  - ... logic: connections between truths – neutral wrt **truth**
  - ... ontology: connections between things – neutral wrt **reality**

(Guarino, 2002) (Masolo et al, 2003)

Descriptive Ontology for Linguistic and Cognitive Engineering

- Strong cognitive/linguistic bias:
  - Descriptive (as opposite to prescriptive) attitude
  - Categories mirror cognition, common sense, and the lexical structure of natural language.
- Emphasis on cognitive invariants
- Categories as conceptual containers: no 'deep' metaphysical implications
- Focus on design rationale to allow easy comparison with different ontological options
- Rigorous, systematic, interdisciplinary approach
- Rich axiomatization
  - 37 basic categories
  - 7 basic relations
  - 80 axioms, 100 definitions, 20 theorems
- Rigorous quality criteria
- Documentation

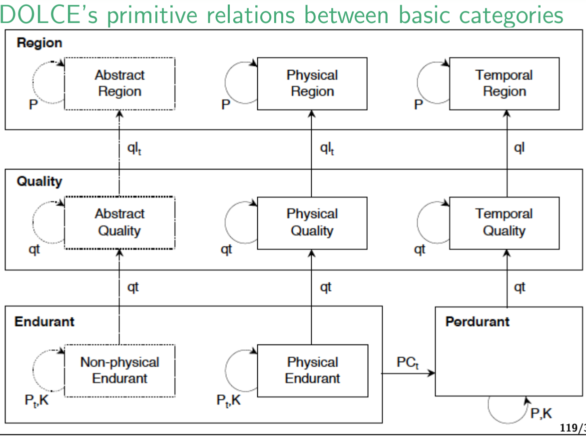
Outline of DOLCE categories



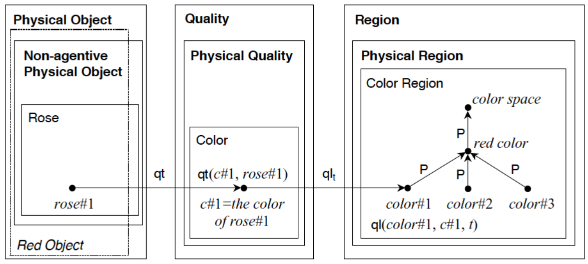
DOLCE's basic relations

- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Constitution
- Participation
- Representation
- Dependence: Specific/generic constant dependence
- Inherence (between a quality and its host)
- Quale
  - Between a quality and its region (immediate, for unchanging entities)
  - Between a quality and its region (temporary, for changing entities)

DOLCE's primitive relations between basic categories

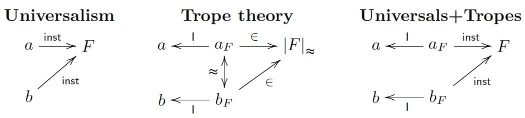


DOLCE's basic relations w.r.t. qualities



Various commitments regarding 'attributes'

- DOLCE: [*PerDurant*/*EnDurant*] –*qt*– *Quality* –*ql*– *Region*
- Options:



- OWL: `DataProperty` with as domain class and range a datatype
  - More compact notation
  - But modelling based on arbitrary (and practical, application) decisions, increasing the chance of incompatibilities and less reusable

DOLCE's basics on universals

(Dd1)	$\mathsf{RG}(\phi) \triangleq \Box\forall x(\phi(x) \to \Box\phi(x))$	(ϕ *is Rigid*)
(Dd2)	$\mathsf{NEP}(\phi) \triangleq \Box\exists x(\phi(x))$	(ϕ *is Non-Empty*)
(Dd3)	$\mathsf{DJ}(\phi,\psi) \triangleq \Box\neg\exists x(\phi(x) \wedge \psi(x))$	(ϕ *and* ψ *are Disjoint*)
(Dd4)	$\mathsf{SB}(\phi,\psi) \triangleq \Box\forall x(\psi(x) \to \phi(x))$	(ϕ *Subsumes* ψ)
(Dd5)	$\mathsf{EQ}(\phi,\psi) \triangleq \mathsf{SB}(\phi,\psi) \wedge \mathsf{SB}(\psi,\phi)$	(ϕ *and* ψ *are Equal*)
(Dd6)	$\mathsf{PSB}(\phi,\psi) \triangleq \mathsf{SB}(\phi,\psi) \wedge \neg\mathsf{SB}(\phi,\psi)$	(ϕ *Properly Subsumes* ψ)
(Dd7)	$\mathsf{L}(\phi) \triangleq \Box\forall\psi(\mathsf{SB}(\phi,\psi) \to \mathsf{EQ}(\phi,\psi))$	(ϕ *is a Leaf*)
(Dd8)	$\mathsf{SBL}(\phi,\psi) \triangleq \mathsf{SB}(\phi,\psi) \wedge \mathsf{L}(\psi)$	(ψ *is a Leaf Subsumed by* ϕ)
(Dd9)	$\mathsf{PSBL}(\phi,\psi) \triangleq \mathsf{PSB}(\phi,\psi) \wedge \mathsf{L}(\psi)$	(ψ *is a Leaf Properly Subsumed by* ϕ)

....

DOLCE's characterisation of categories

Physical Object

(Ad32)* $\mathsf{GK}(SC, SAG)$

(Ad30)* $\mathsf{GK}(NAPO, M)$

(Ad70)* $\mathsf{OGD}(F, NAPO)$

(Ad71)* $\mathsf{OSD}(MOB, APO)$

(Ad72)* $\mathsf{OGD}(SAG, APO)$

Feature

(Ad70)* $\mathsf{OGD}(F, NAPO)$

Non-physical Endurant

(Ad12)* $\mathsf{P}(x,y,t) \to (NPED(x) \leftrightarrow NPED(y))$

(Ad22)* $\mathsf{K}(x,y,t) \to (NPED(x) \leftrightarrow NPED(y))$

(Ad41)* $\mathsf{qt}(x,y) \to (AQ(x) \leftrightarrow (AQ(y) \vee NPED(y)))$

(Ad48)* $AQ(x) \to \exists! y(\mathsf{qt}(x,y) \wedge NPED(y))$

(Ad51)* $NPED(x) \to \exists\phi, y(\mathsf{SBL}(AQ,\phi) \wedge \mathsf{qt}(\phi,y,x))$

(Ad74)* $\mathsf{OD}(NPED, PED)$

... etc...

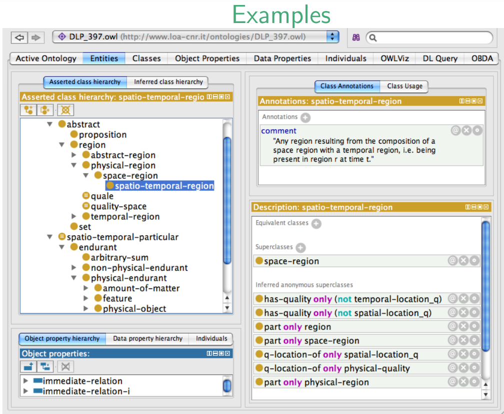
Can all that be used?

- DOLCE in KIF
- DOLCE in OWL:
  - DOLCE-Lite: simplified translations of Dolce2.0
  - Does *not* consider: modality, temporal indexing, relation composition
  - Different names are adopted for relations that have the same name but different arities in the FOL version
  - Some commonsense concepts have been added as examples
- DOLCE-2.1-Lite-Plus version includes some modules for Plans, Information Objects, Semiotics, Temporal relations, Social notions (collectives, organizations, etc.), a Reification vocabulary, etc.

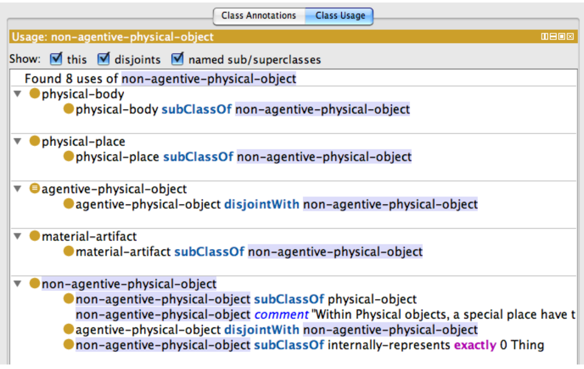
DLP3971

- Several Modules for (re)use: DOLCE-Lite, SocialUnits, SpatialRelations, ExtendedDnS, and others
- Still rather complex to understand (aside from using OWL-DL): Full DOLCE-Lite-Plus with 208 classes, 313 object properties, etc (check the "Active ontology" tab in Protégé) and basic DOLCE-Lite 37 classes, 70 object properties etc (in $\mathcal{SHI}$)
- Time for a DOLCE-Lite ultra-"ultralight"? e.g. for use with OWL 2 QL or OWL 2 EL
  - Current DOLCE Ultra Lite—DUL—uses friendly names and comments for classes and properties, has simple restrictions for classes, and includes into a unique file the main parts of DOLCE, D&S and other modules of DOLCE Lite+
  - BUT... is still in OWL-DL (OWL-Lite+Disjointness)
- `http://wiki.loa-cnr.it/index.php/LoaWiki:Ontologies`

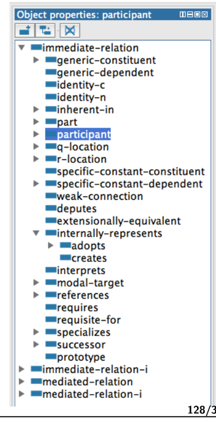
Examples



Examples



Comment: "The immediate relation holding between endurants and perdurants (e.g. in 'the car is running').Participation can be constant (in all parts of the perdurant, e.g. in 'the car is running'), or temporary (in only some parts, e.g. in 'I'm electing the president').A 'functional' participant is specialized for those forms of participation that depend on the nature of participants, processes, or on the intentionality of agentive participants. Traditional 'thematic role' should be mapped to functional participation.For relations holding between participants in a same perdurant, see the co-participates relation."



BFO Overview

- Ontology as reality representation
- Aims at reconciling the so-called three-dimensionalist and four-dimensionalist views
  - A Snap ontology of endurants which is reproduced at each moment of time and is used to characterize static views of the world
  - Span ontology of happenings and occurrents and, more generally, of entities which persist in time by perduring
  - Endurants (Snap) or perdurants (Span)
- Limited granularity
- Heavily influenced by parthood relations, boundaries, dependence

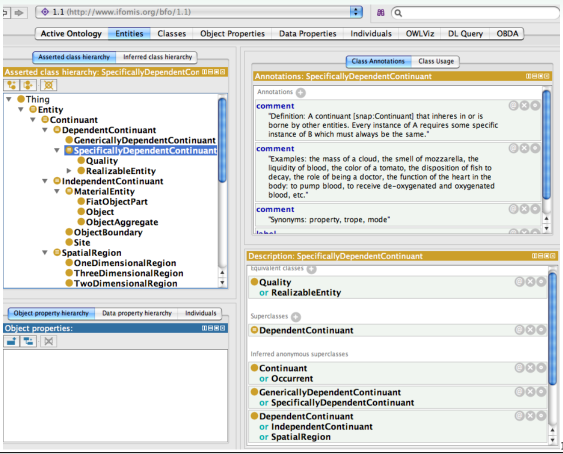
Overview

- BFO 1.1 in OWL with 39 classes, no object or data properties, in $\mathcal{ALC}$.
- There is a `bfo-ro.owl` to integration relations of the Relation Ontology with BFO (extensions under consideration)
- Version in Isabelle (mainly part-wholes, but not all categories)
- Version in OBO (the original Gene Ontology format, with limited, but expanding, types of relationships)
- Version in Prover9 (first order logic model checker and theorem prover)

BFO Taxonomy

```
bfo:Entity
  snap:Continuant
    snap:DependentContinuant
      snap:GenericalyDependentContinuant
      snap:SpecificalyDependentContinuant
        snap:Quality
        snap:RealizableEntity
          snap:Disposition
          snap:Function
          snap:Role
    snap:IndependentContinuant
      snap:MaterialEntity
          snap:Object
          snap:FiatObjectPart
          snap:ObjectAggregate
      snap:ObjectBoundary
      snap:Site
    snap:SpatialRegion
      snap:ZeroDimensionalRegion
      snap:OneDimensionalRegion
      snap:TwoDimensionalRegion
      snap:ThreeDimensionalRegion

  span:Occurrent
    span:ProcessualEntity
      span:Process
      span:ProcessBoundary
      span:FiatProcessPart
      span:ProcessAggregate
      span:ProcessualContext
    span:SpatiotemporalRegion
      span:ConnectedTemporalRegion
        span:SpatiotemporalInstant
        span:SpatiotemporalInterval
      span:ScatteredSpatiotemporalRegion
    span:TemporalRegion
      span:ConnectedSpatiotemporalRegion
        span:TemporalInstant
        span:TemporalInterval
      span:ScatteredTemporalRegion
```

BFO Core

- A non-extensional temporal mereology with collections, sums, and universals
- BFO as a collection of smaller theories
  - EMR, QSizeR, RBG, QDiaSizeR, ..., Adjacency, Collections, SumsPartitions, Universals, Instantiation, ExtensionsOfUniversals, PartonomicInclusion, UniversalParthood
- Reference material `http://www.ifomis.org/bfo/fol` and `http://www.acsu.buffalo.edu/~bittner3/Theories/BFO/`

Section of one of the sub-theories in BFO Core

theory *UniversalParthood*

imports *ExtensionsOfUniversals PartonomicInclusion*

begin

consts

UPt1 :: *Un* => *Un* => *Ti* => *o*
UPt2 :: *Un* => *Un* => *Ti* => *o*
UPt12 :: *Un* => *Un* => *Ti* => *o*

UP1 :: *Un* => *Un* => *o*
UP2 :: *Un* => *Un* => *o*
UP12 :: *Un* => *Un* => *o*

defs

UPt1-def: *UPt1(c,d,t) == (ALL x. (Inst(x,c,t) --> (EX y. (Inst(y,d,t) & P(x,y,t)))))*
UPt2-def: *UPt2(c,d,t) == (ALL y. (Inst(y,d,t) --> (EX x. (Inst(x,c,t) & P(x,y,t)))))*
UPt12-def: *UPt12(c,d,t) == UPt1(c,d,t) & UPt2(c,d,t)*

UP1-def: *UP1(c,d) == (ALL t. UPt1(c,d,t))*
UP2-def: *UP2(c,d) == (ALL t. UPt2(c,d,t))*
UP12-def: *UP12(c,d) == (ALL t. UPt12(c,d,t))*

A relation ontology

- What are the 'core' and primitive relations necessary to develop a domain ontology?
- Do we need a *separate* ontology for relations, or integrated in a foundational ontology?
- Philosophers do not agree on the answers, but the modellers and engineers need agreement to facilitate interoperability among ontologies

The Relation Ontology

- Definitions for *is_a*, *part_of*, *integral_part_of*, *proper_part_of*, *located_in*, *contained_in*, *adjacent_to*, *transformation_of*, *derives_from*, *preceded_by*, *has_participant*, *has_agent*, *instance_of*
- Proposed extensions under consideration, among others:
  - Relations between generically dependent continuants and specifically dependent continuants (a.o., concretizes, *has_quality*, *has_function*, ...)
  - A relation between a process and a process or quality (*regulates*)
  - Refinements on *derived_from*
  - Measurements (*has_value*, *of_dimension*, ...)

Ontologies and choices

- Other more or less used foundational ontologies, a.o.:
  - GFO
  - SUMO
  - OCHRE
  - ...
- Within WonderWeb project: a (future) aim to develop a library of foundational ontologies with mappings between them: choose your pet ontology and be interoperable with the others
- Exercise: examine DolceliteBFOinDLandMSyntax.pdf (or their respective OWL files) and spot commonalities and differences between DOLCE and BFO (or any two other foundational ontologies)

Some questions and problems (not exhaustive...)[4]

- Is a tunnel part of the mountain?
- What is the difference, if any, between how `Cell nucleus` and `Cell` are related and how `Receptor` and `Cell wall` are related?
- And w.r.t. `Brain part of Human` and/versus `Hand part of Boxer`? (assuming boxers must have their own hands)
- A classical example: hand is part of musician, musician part of orchestra, but clearly, the musician's hands are not part of the orchestra. Is part-of then not transitive, or is there a problem with the example?

[4] The following slides are based on the tutorial given at Meraka [http://www.meteck.org/files/PartspresMOWS08.pdf], which does have the references to the related works.

Analysis of the issues from diverse angles

- Mereological theories (Varzi, 2004), usage & extensions (e.g. mereotopology, relation with granularity, set theory)
- Early attempts with direct parthood, SEP triples, and other outstanding issues, some still remaining
- Cognitive & linguistic issues from meronymy
- Usage in conceptual modelling and ontology engineering
- Subject domains: thus far, mainly geo, bio, medicine

Ground Mereology

Reflexivity (everything is part of itself)

$$\forall x(part\_of(x,x)) \quad (1)$$

Antisymmetry (two distinct things cannot be part of each other, or: if they are, then they are the same thing)

$$\forall x,y((part\_of(x,y) \land part\_of(y,x)) \rightarrow x = y) \quad (2)$$

Transitivity (if x is part of y and y is part of z, then x is part of z)

$$\forall x,y,z((part\_of(x,y) \land part\_of(y,z)) \rightarrow part\_of(x,z)) \quad (3)$$

Proper parthood

$$\forall x,y(proper\_part\_of(x,y) \equiv part\_of(x,y) \land \neg part\_of(y,x)) \quad (4)$$

Ground Mereology

Proper parthood

$$\forall x,y(proper\_part\_of(x,y) \equiv part\_of(x,y) \land \neg part\_of(y,x)) \quad (5)$$

Asymmetry (if x is part of y then y is not part of x)

$$\forall x,y(part\_of(x,y) \rightarrow \neg part\_of(y,x)) \quad (6)$$

Irreflexivity (x is not part of itself)

$$\forall x \neg(part\_of(x,x)) \quad (7)$$

Defining other relations with *part_of*

Overlap (x and y share a piece z)

$$\forall x,y(overlap(x,y) \equiv \exists z(part\_of(z,x) \land part\_of(z,y))) \quad (8)$$

Underlap (x and y are both part of some z)

$$\forall x,y(underlap(x,y) \equiv \exists z(part\_of(x,z) \land part\_of(y,z))) \quad (9)$$

Over- & undercross (over/underlap but not part of)

$$\forall x,y(overcross(x,y) \equiv overlap(x,y) \land \neg part\_of(x,y)) \quad (10)$$

$$\forall x,y(undercross(x,y) \equiv underlap(x,y) \land \neg part\_of(y,x)) \quad (11)$$

Proper overlap & Proper underlap

$$\forall x,y(p\_overlap(x,y) \equiv overcross(x,y) \land overcross(y,x)) \quad (12)$$

$$\forall x,y(p\_underlap(x,y) \equiv undercross(x,y) \land undercross(y,x)) \quad (13)$$

- With x as part, what to do with the remainder that makes up y?
  - Weak supplementation: every proper part must be supplemented by another, disjoint, part. **MM**
  - Strong supplementation: if an object fails to include another among its parts, then there must be a remainder. **EM**
- Problem with EM: non-atomic objects with the same proper parts are identical, because of this (extensionality principle), but sameness of parts may not be sufficient for identity E.g.: two objects can be distinct purely based on arrangement of its parts, differences statue and its marble (multiplicative approach)

General Extensional Mereology

- Strong supplementation [EM]

$$\neg part\_of(y,x) \rightarrow \exists z(part\_of(z,y) \wedge \neg overlap(z,x)) \quad (14)$$

- And add unrestricted fusion [GEM]. Let ϕ be a property or condition, then for every satisfied ϕ there is an entity consisting of all entities that satisfy ϕ. [5] Then:

$$\exists x\phi \rightarrow \exists z\forall y(overlap(y,z) \leftrightarrow \exists x(\phi \wedge overlap(y,x))) \quad (15)$$

- Note that in EM and upward we have identity, from which one can prove acyclicity for ppo
- There are more mereological theories, and the above is not uncontested (more about that later)

[5] Need to refer to classes, but desire to stay within FOL. Solution: axiom schema with only predicates or open formulas

Relations between common mereological theories

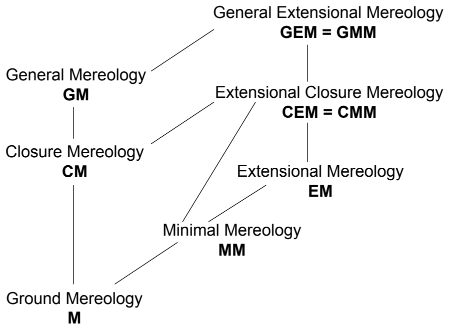


Fig. 1: Hasse diagram of mereological theories; from weaker to stronger, going uphill (after [44]).

Can any of this be represented in a decidable fragment of first order logic for use in ontologies and (scalable) software implementations?

Things are improving...

- Early days (90s) and simplest options: DL-role `R` as `partof`, or `has-part` added as primitive role as $\succeq$, model it as the transitive closure of a parthood relation (16) and define e.g. `Car` as having wheels that in turn have tires (17):

$$\succeq \doteq (\texttt{primitive-part})* \quad (16)$$

$$\texttt{Car} \doteq \exists \succeq .(\texttt{Wheel} \sqcap \exists \succeq .\texttt{Tire}) \quad (17)$$

Then $\texttt{Car} \sqsubseteq \exists \succeq .\texttt{Tire}$
- SEP triples with $\mathcal{ALC}$
- What $\mathcal{SHIQ}$ fixes cf. $\mathcal{ALC}$: Transitive roles, Inverse roles (to have both part-of and has-part), Role hierarchies (e.g. for subtypes of part-of), qualified Number restrictions (e.g. to represent that a bycicle has-part 2 wheels)
- Build-your-own DL-language

What we can(not) implement now with DL-based ontology languages

Table: Properties of parthood and proper parthood compared to their support in $\mathcal{DLR}_\mu$, $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$. *: properties of the parthood relation (in M); ‡: properties of the proper parthood relation (in M).

Language ⇒ **Feature** ⇓	$\mathcal{DLR}_\mu$	$\mathcal{SHOIN}$ (∼ OWL-DL)	$\mathcal{SROIQ}$ (∼ OWL 2 DL)	DL-Lite$_A$ (∼ OWL 2 QL)
Reflexivity *	+	–	+	–
Antisymmetry *	–	–	–	–
Transitivity * ‡	+	+	+	–
Asymmetry ‡	+	+	+	+
Irreflexivity ‡	+	–	+	–
Acyclicity	+	–	–	–

Definitions in OBO Relations Ontology

- Instance-level relations
  - c **part_of** c_1 at t - a primitive relation between two continuant instances and a time at which the one is part of the other
  - p **part_of** p_1, r **part_of** r_1 - a primitive relation of parthood, holding independently of time, either between process instances (one a subprocess of the other), or between spatial regions (one a subregion of the other)
  - c **contained_in** c_1 at $t \triangleq c$ **located_in** c_1 at t and not c **overlap** c_1 at t
  - c located_in r at t - a primitive relation between a continuant instance, a spatial region which it occupies, and a time

Definitions in OBO Relations Ontology

- Class-level relations
  - C *part_of* $C_1 \triangleq$ for all c, t, if Cct then there is some c_1 such that C_1c_1t and c **part_of** c_1 **at** t.
  - P *part_of* $P_1 \triangleq$ for all p, if Pp then there is some p_1 such that: P_1p_1 and p **part_of** p_1.
  - C *contained_in* $C_1 \triangleq$ for all c, t, if Cct then there is some c_1 such that: C_1c_1t and c **contained_in** c_1 at t
- Need to commit to a foundational ontology. Recently, linked to BFO http://obofoundry.org/ro/#mappings (test release)
- Same labels, different relata and only a textual constraint: Label the relations differently

Linguistic use of part-whole relations (meronymy)

- Part of?
  - ⋆ Centimeter part of Decimeter
  - ⋆ Decimeter part of Meter
  - — *therefore* Centimeter part of Meter
  - ⋆ Meter part of SI
  - — but *not* Centimeter part of SI
- Transitivity?
  - ⋆ Person part of Organisation
  - ⋆ Organisation located in Bolzano
  - — therefore Person located in Bolzano?
  - — but *not* Person part of Bolzano

Linguistic use of part-whole relations (meronymy)

- Part of?
  - ⋆ Centimeter part of Decimeter
  - ⋆ Decimeter part of Meter
  - — *therefore* Centimeter part of Meter
  - ⋆ Meter part of SI
  - — but *not* Centimeter part of SI
- Transitivity?
  - ⋆ Person member of Organisation
  - ⋆ Organisation located in Bolzano
  - — therefore Person located in Bolzano?
  - — but *not* Person member of Bolzano

Linguistic use of part-whole relations

- Which part of?
  - ⋆ CellMembrane structural part of RedBloodCell
  - ⋆ RedBloodCell part of Blood
  - — but *not* CellMembrane structural part of Blood
  - ⋆ Receptor structural part of CellMembrane
  - — *therefore* Receptor structural part of RedBloodCell

Linguistic use of part-whole relations

- Which part of?
  - ⋆ CellMembrane structural part of RedBloodCell
  - ⋆ RedBloodCell contained in? Blood
  - — but *not* CellMembrane structural part of Blood
  - ⋆ Receptor structural part of CellMembrane
  - — *therefore* Receptor structural part of RedBloodCell

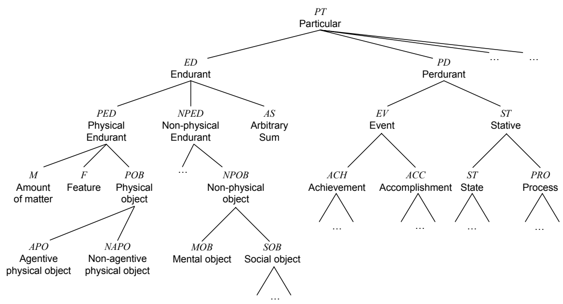
Addressing the issues

- Efforts to disambiguate this confusion; e.g. an informal taxonomy by Winston et al (1987), list of 6 types motivated by UML conceptual modeling (Odell) ontology-inspired conceptual modelling (Guizzardi)
- Location, containment, membership of a collective, quantities of a mass
- Relatively well-settled debate on transitivity, or not

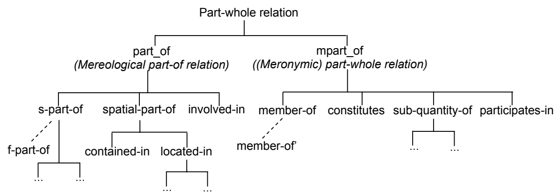
Overview

- Mereological *part_of* (and subtypes) versus ‘other’ part-whole relations
- Categories of object types of the part-whole relation changes
- Structure these relations by (non/in)transitivity and kinds of relata
- Simplest mereological theory, **M**.
- Commit to a foundational ontology: DOLCE (though one also could choose, a.o., BFO, OCHRE, GFO, ...)

DOLCE categories



Part-whole relations



Part-whole relations

“member-bunch”, collective nouns (e.g. `Herd`, `Orchestra`) with their members (`Sheep`, `Musician`)

$$\forall x,y(member\_of_n(x,y) \triangleq mpart\_of(x,y) \land (POB(x) \lor SOB(x)) \land SOB(y))$$

“material-object”, that what something is made of (e.g., `Vase` and `Clay`)

$$\forall x,y(constitutes_{it}(x,y) \equiv constituted\_of_{it}(y,x) \triangleq mpart\_of(x,y) \land POB(y) \land M(x))$$

Part-whole relations

“quantity-mass”, “portion-object”, relating a smaller (or sub) part of an amount of matter to the whole. Two issues (glass of wine & bottle of wine vs. `Salt` as subquantity of `SeaWater`)

$$\forall x,y(sub\_quantity\_of_n(x,y) \triangleq mpart\_of(x,y) \land M(x) \land M(y))$$

“noun-feature/activity”, entity participates in a process, like `Enzyme` that participates in `CatalyticReaction`

$$\forall x,y(participates\_in_{it}(x,y) \triangleq mpart\_of(x,y) \land ED(x) \land PD(y))$$

Part-whole relations

processes and sub-processes (e.g. `Chewing` is involved in the grander process of `Eating`)

$$\forall x, y(involved\_in(x, y) \triangleq part\_of(x, y) \wedge PD(x) \wedge PD(y))$$

Object and its 2D or 3D region, such as `contained_in(John's address book, John's bag)` and `located_in(Pretoria, South Africa)`

$$\forall x, y(contained\_in(x, y) \triangleq part\_of(x, y) \wedge R(x) \wedge R(y) \wedge \exists z, w(has\_3D(z, x) \wedge has\_3D(w, y) \wedge ED(z) \wedge ED(w)))$$

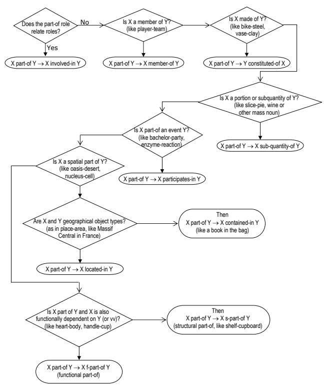
$$\forall x, y(located\_in(x, y) \triangleq part\_of(x, y) \wedge R(x) \wedge R(y) \wedge \exists z, w(has\_2D(z, x) \wedge has\_2D(w, y) \wedge ED(z) \wedge ED(w)))$$

$$\forall x, y(s\_part\_of(x, y) \triangleq part\_of(x, y) \wedge ED(x) \wedge ED(y))$$

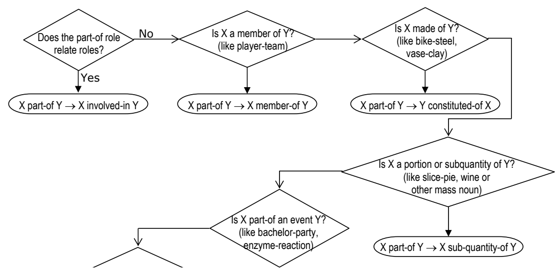
Using the taxonomy of part-whole relations

- Representing it correctly in ontologies and conceptual data models
  - Decision diagram
  - Using the categories of the foundational ontology
  - Examples
  - *Software application* that simplifies all that
- Reasoning with a taxonomy of relations
  - The *RBox reasoning service* to pinpoint errors

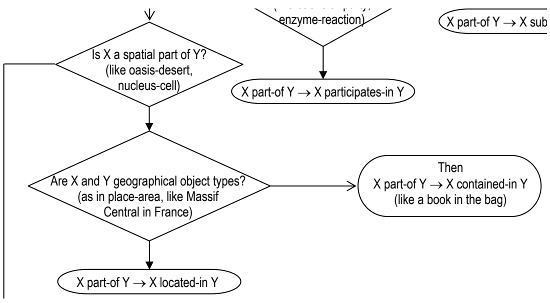
Decision diagram



Decision diagram



Decision diagram



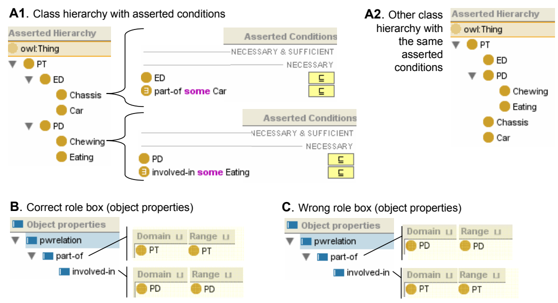
Using DOLCE's categories

- The participating objects instantiate some category (*ED*, *PD*, etc)
- Given the formalization, one immediately can exclude/identify appropriate relations, taking a shortcut in the decision diagram
  - E.g.: *Chewing* and *Eating* are both a kind of (a subtype of) *PD*, hence *involved_in*
  - E.g.: *Alcohol* and *Wine* are both mass nouns, or *M*, hence *sub_quantity_of*
- Demo of OntoPartS

Requirements for reasoning over the hierarchy

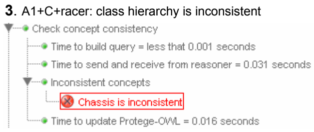
- Represent at least Ground Mereology,
- Express ontological categories and their taxonomic relations,
- Having the option to represent transitive and intransitive relations, and
- Specify the domain and range restrictions (/relata/entity types) for the classes participating in a relation.

Current behaviour of reasoners

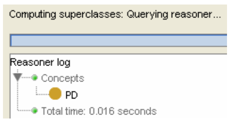


Current behaviour of reasoners

1. A1+B+racer: *ontology OK*
2. A2+B+racer: *ontology OK*
3. A1+C+racer: class hierarchy is inconsistent



4. A2+C+racer: Chassis reclassified as PD



The *RBox Compatibility* service – definitions

Definition (Domain and Range Concepts)

Let R be a role and $R \sqsubseteq C_1 \times C_2$ its associated Domain & Range axiom. Then, with the symbol D_R we indicate the *User-defined Domain* of R—i.e., $D_R = C_1$—while with the symbol R_R we indicate the *User-defined Range* of R—i.e., $R_R = C_2$.

Definition (RBox Compatibility)

For each pair of roles, R, S, such that $\langle \mathcal{T}, \mathcal{R} \rangle \models R \sqsubseteq S$, check:

Test 1. $\langle \mathcal{T}, \mathcal{R} \rangle \models D_R \sqsubseteq D_S$ and $\langle \mathcal{T}, \mathcal{R} \rangle \models R_R \sqsubseteq R_S$;

Test 2. $\langle \mathcal{T}, \mathcal{R} \rangle \not\models D_S \sqsubseteq D_R$;

Test 3. $\langle \mathcal{T}, \mathcal{R} \rangle \not\models R_S \sqsubseteq R_R$.

An RBox is said to be compatible iff `Test 1` and (*2* or *3*) hold for all pairs of role-subrole in the RBox.

The *RBox Compatibility* service – behaviour

- If `Test 1` does not hold: warning that domain & range restrictions of either R or S are in conflict with the role hierarchy proposing either
  (i) To change the role hierarchy or
  (ii) To change domain & range restrictions or
  (iii) If the test on the domains fails, then propose a new axiom $R \sqsubseteq D'_R \times R_R$, where $D'_R \equiv D_R \sqcap D_S$[6], which subsequently has to go through the RBox compatibility service (and similarly when `Test 1` fails on range restrictions).

[6] The axiom $C_1 \equiv C_2$ is a shortcut for the axioms: $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$.

The *RBox Compatibility* service – behaviour

- If `Test 2` and `Test 3` fail: warn that R cannot be a proper subrole of S but that the two roles can be equivalent. Then, either:
  (a) Accept the possible equivalence between the two roles or
  (b) Change domain & range restrictions.
- Ignoring all warnings is allowed, too

Extensions in various directions

- Mereotopology, with location, GIS, Region Connection Calculus (http://www.comp.leeds.ac.uk/qsr/rcc.html)
- Mereogeometry
- Mereology and/vs granularity
- Temporal aspects of part-whole relations

Knowledge and Google & AfriGIS

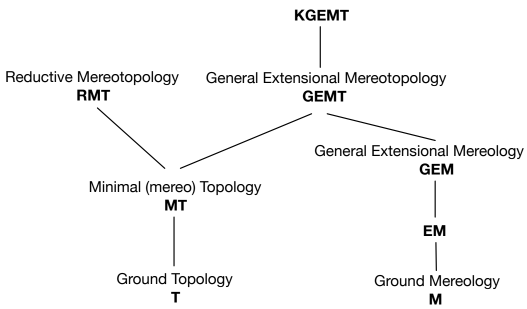
Knowledge and Google & AfriGIS

- How can we represent
  - The Kruger Park *overlaps* with South Africa
  - Durban is a *tangential proper part* of South Africa
  - Gauteng is a *non-tangential proper part* of South Africa
  - Botswana is *connected to* South Africa (do they *share* a border?)
  - Lesotho is *spatially located within* the area of South Africa (but not part of)?
- Can we do all that with mereology? Use only spatial relations? Combining mereo+spatial?

Mereology with spatial notions

- Another primitive: Connected, which is reflexive and symmetric
- More and more expressive theories, e.g.:
  - T: $C(x,x)$ and $C(x,y) \rightarrow C(y,x)$
  - MT: T and $P(x,y) \rightarrow E(x,y)$ where E is enclosure $(E(x,y) =_{def} \forall z(C(z,x) \rightarrow C(z,y)))$
- Two primitives, P and C, or *part* in terms of C?
  - $P =_{def} \forall z(C(z,x) \rightarrow C(z,y))$
- or perhaps "x and y are connected parts of z" as primitive, $CP(x,y,z)$, then:
  $P(x,y) =_{def} \exists z \; CP(x,z,y)$ and
  $C(x,y) =_{def} \exists z \; CP(x,y,z)$

Some of the mereo- and topological theories



Note: one can add explicit variations with Atom/Atomless and Boundary/Boundaryless

Rationale

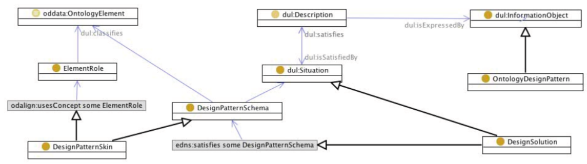
- It is hard to reuse only the "useful pieces" of a comprehensive (foundational) ontology, and the cost of reuse may be higher than developing a new ontology from scratch
- Need for small (or cleverly modularized) ontologies with explicit documentation of design rationales, and best reengineering practices
- Hence, in analogy to software design patterns: **ontology design patterns**
- ODPs summarize the good practices to be applied within design solutions
- ODPs keep track of the design rationales that have motivated their adoption

content of slides based on Presutti et al, 2008

ODP definition

- An ODP is an information object
- A design pattern schema is the description of an ODP, including the roles, tasks, and parameters needed in order to solve an ontology design issue
- *An ODP is a modeling solution to solve a recurrent ontology design problem. It is an Information Object that expresses a Design Pattern Schema (or skin) that can only be satisfied by DesignSolutions. Design solutions provide the setting for Ontology Elements that play some ElementRole(s) from the schema.* (Presutti et al, 2008)

ODP types



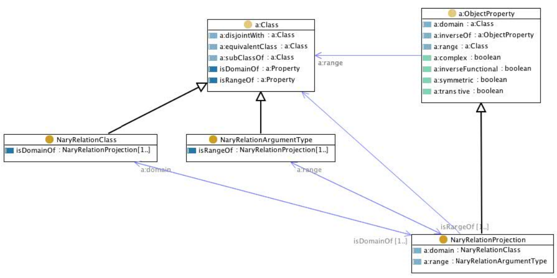
Types of Patterns

- Six families of ODPs: Structural OPs, Correspondence OPs, Content OPs (CPs), Reasoning OPs, Presentation OPs, and Lexico-Syntactic OPs
- CPs can be distinguished in terms of the domain they represent
- Correspondence OPs (for reengineering and mappings—next lecture)
- Reasoning OPs are typical reasoning procedures
- Presentation OPs relate to ontology usability from a user perspective; e.g., we distinguish between Naming OPs and Annotation OPs
- Lexico-Syntactic OP are linguistic structures or schemas that permit to generalize and extract some conclusions about the meaning they express

Structural OPs

- Logical OPs:
  - Are compositions of logical constructs that solve a problem of expressivity in OWL-DL (and, in cases, also in OWL 2 DL)
  - Only expressed in terms of a logical vocabulary, because their signature (the set of predicate names, e.g. the set of classes and properties in an OWL ontology) is empty
  - Independent from a specific domain of interest
  - **Logical macros** compose OWL DL constructs; e.g. the universal+existential OWL macro
  - **Transformation patterns** translate a logical expression from a logical language into another; e.g. n-aries

Example: n-ary relation Logical OP



Architectural OPs

- Architectural OPs are defined in terms of composition of Logical OPs that are used in order to affect the overall shape of the ontology; i.e., an Architectural OP identifies a composition of Logical OPs that are to be exclusively used in the design of an ontology
- Examples of Architectural OPs are: Taxonomy, Modular Architecture, and Lightweight Ontology
- E.g., **Modular Architecture** Architectural OP consists of an ontology network, where the involved ontologies play the role of modules, which are connected by the *owl:import* operation with one root ontology that imports all the modules

Lexico-Syntactic OPs

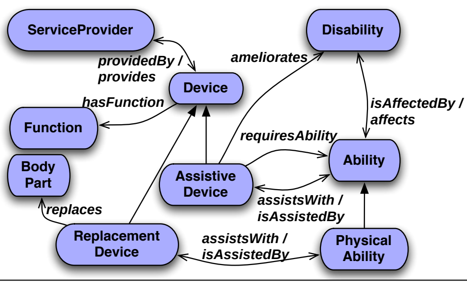
- linguistic structures or schemas that consist of certain types of words following a specific order and that permit to generalize and extract some conclusions about the meaning they express; *verbalisation* patterns
- E.g., "subClassOf" relation, NP<subclass> be NP<superclass>, a Noun Phrase should appear before the verb—represented by its basic form or lemma, be in this example—and the verb should in its turn be followed by another Noun Phrase
- Other Lexical OPs provided for OWL's equivalence between classes, object property, subpropertyOf relation, datatype property, existential restriction, universal restriction, disjointness, union of classes
- Mainly for English language only, thus far
- Similar to idea of ORM's verbalization templates

How to create an ODP

- See chapter 3 of (Presutti et al., 2008)
- Where do ODPs come from (section 3.4—in part: legacy sources, which we deal with in the next lecture)
- Annotation schema
- How to use them
- Content Ontology Design Anti-pattern (AntiCP)

Sample exercise: an ODP for the ADOLENA ontology?

- Novel Abilities and Disabilities OntoLogy for ENhancing Accessibility: ADOLENA
- Can this be engineered into an ODP? If so, which type(s), how, what information is needed to document an ODP?



Summary

Part IV

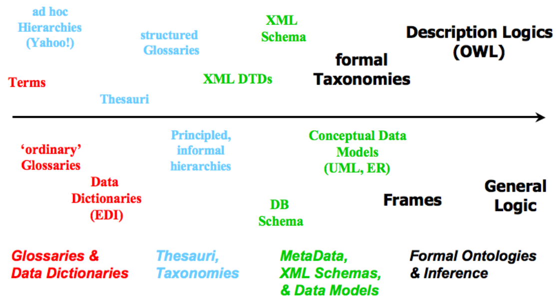
Bottom-up ontology development

Outline

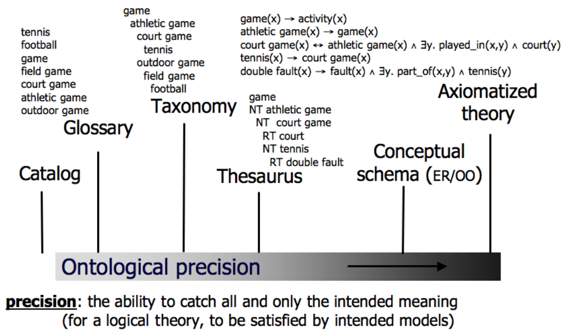
Bottom-up

- From *some* seemingly suitable legacy representation to an OWL ontology
  - Database reverse engineering
  - Conceptual model (ER, UML)
  - Frame-based system
  - OBO format
  - Thesauri
  - Formalizing biological models
  - Excel sheets
  - Text mining, machine learning, clustering
  - etc...

A few languages



Levels of ontological precision



(from Gangemi, 2004)

Examples: OBO and Protégé-frames

- OBO in OWL 2 DL
  - OBO is a Directed Acyclic Graph (with is_a, part_of, etc. relationships)
  - with some extras (a.o., date, saved by, remark)
  - and 'work-arounds' (not-necessary and inverse-necessary) and non-mappable things (antisymmetry)
  - There are several OBO-in-OWL mappings, some more comprehensive than others
  - e.g. FMA-Lite

Examples: OBO and Protégé-frames

- Frames (as in Protégé) into OWL-DL (see Zhang & Bodenreider, 2004), and its problems doing that to the FMA
  - Not a formal transformation
  - Slot values generally correspond to necessary conditions—so they took a first guess to define an anatomical entity as the sum of its parts
  - Global axioms dropped (with an eye on the reasoner)
  - After the conversion of the 39,337 classes and 187 slots from FMA in Protégé (ignoring laterality distinctions), FMAinOWL contains 39,337 classes, 187 properties and 85 individuals
  - Additional optimizations: optimizing domains and subClassOf axioms
  - But still caused Racer to fail to reason over the whole file; restricting properties further obtained results

General considerations for RDBMSs

- Set aside of data duplication, violations of integrity constraints, hacks, outdated imports from other databases, outdated conceptual data models
- Some data in the DB—mathematically instances—actually assumed to be concepts/universals/classes
- 'impedance mismatch' DB values and ABox objects
- ⇒ instances-but-actually-concepts-that-should-become-OWL-classes and real-instances-that-should-become-OWL-instances

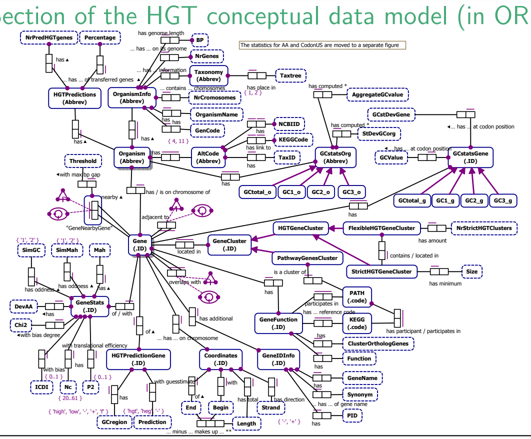
General considerations for RDBMSs

- Reuse/reverse engineer the physical DB schema
- Reuse conceptual data model (in ER, EER, UML, ORM, ...)
- But,
  - Assumes there was a fully normalised conceptual data model,
  - Denormalization steps to flatten the database structure, which, if simply reverse engineered, ends up in the ontology as a class with umpteen attributes
  - Minimal (if at all) automated reasoning with it
- Redo the normalization steps to try to get some structure back into the conceptual view of the data?
- Add a section of another ontology to brighten up the 'ontology' into an ontology?
- Establish some mechanism to keep a 'link' between the terms in the ontology and the source in the database?

Manual Extraction

- Most database are not neat as assumed in the 'Automatic Extraction of Ontologies' (e.g., denormalised)
- Then what?
  - Reverse engineer the database to a conceptual data model
  - Choose an ontology language for your purpose
- Example: the HGT-DB about horizontal gene transfer (the same holds for the database behind ADOLENA)

Section of the HGT conceptual data model (in ORM 2)



Manual mapping to *DL-Lite*$_{\mathcal{A}}$

- Basic statistics:
  - 38 classes
  - 34 object properties of which 17 functional
  - 55 data properties of which 47 functional
  - 102 subclass axioms
- Subsequently used for Ontology-Based Data Access

Automatic Extraction of Ontologies

- Examples
  - Lina Lubyte & Sergio Tessaris's presentation of the DEXA'09 paper
  - Reverse engineering from DB to ORM model with, e.g., VisioModeler v3.1 or NORMA

Natural language and ontologies

- Using ontologies to improve NLP
  - To enhance precision and recall of queries
  - To enhance dialogue systems
  - To sort literature results
  - To navigate literature (linked data)
- Using NLP to develop ontologies (TBox)
  - Searching for candidate terms and relations: Ontology learning (today; ref Alexopoulou et al, 2008)
- Using NLP to populate ontologies (ABox)
  - Document retrieval enhanced by lexicalised ontologies
  - Biomedical text mining (today; ref Witte et al, 2007)
- Natural language generation from a formal language

A few notes on the nature of relations

- Early ideas were put forward by Williamson85 and have been elaborated on and structured in Fine00, Inwagen06, Leo08, and Cross02[7]
- Three different ontological commitments[8] about relations and relationships, which are, in Fine's terminology, the *standard view*, the *positionalist*, and the *anti-positionalist* commitment

[7] Full references in Keet, C.M. Positionalism of relations and its consequences for fact-oriented modelling. *Proc. of ORM'09, OTM Workshops*, Springer, LNCS 5872, 735-744.

[8] well, different people are convinced about the nature of the relation in reality; it does not exclude the possibility that maybe the corresponding different formalisations have equivalence-preserving transformations between them and admit the exact same models (if on assumes a model-theoretic semantics)).

The 'standard view' commitment

- Relies on linguistics and the English language in particular
- Take the fact *John loves Mary*, then one could be led to assume that *loves* is the name of the relation and *John* and *Mary* are the objects participating in the relation
- Then *Mary loves John* is not guaranteed to have the same truth value as the former fact—changing the verb does, i.e., *Mary is loved by John*
- We (seem to) have two relations, *loves* and its inverse *is loved by*

Problems with the 'standard view' (1/2)

- First, generally, for names *a* and *b*, *a loves b* holds iff what *a* denotes (in the reality we aim to represent) loves what *b* denotes.
- *John loves Mary* is not about language but about John loving Mary, so John and Mary are non-linguistic; cf. *'cabeza' translates into 'head'*
- Then, that John loves Mary and Mary is being loved by John refer to only one state of affairs between John and Mary
- Why should we want, let alone feel the need, to have *two relations* to describe it?

Toward the 'positionalist' commitment

- Designate the two aforementioned facts to be relational expressions and not to let the verb used in the fact automatically also denote the name of the relation
- Then we can have many relational expressions standing in for the single relation that captures the state of affairs between John and Mary
- In analogy, we can have many relational expressions for one relationship at the type level

Problems with the 'standard view' (2/2)

- Second, the specific order of the relation: changing the order does not mean the same for verbs that indicate an asymmetric relation; different for some other languages.
- Consider *John kills the dragon*. In Latin we have:
  Johannus anguigenam caedit, or
  anguigenam caedit Johannus, or
  Johannus caedit anguigenam,
  which all refer to the same state of affairs
- But *Johannum anguigena caedit* is a different story alltogether
- Likewise for *John loves Mary* and *Johannus Mariam amat* versus *Johannum Maria amat*.

Toward the 'positionalist' commitment

- A linguistic version of *argument places* (roles) thanks to the nominative and the accusative that are linguistically clearly indicated
- The order of the argument places is not relevant for the relation itself
- English without such declensions that change the terms so as to disambiguate the meaning of a relational expression
- *Inverses for seemingly asymmetrical relations necessarily exist in reality and descriptions of reality in English, but not in other languages even when they represent the same state of affairs???*
- Asymmetric **relational expressions**, but this does not imply that the **relation** it verbalises is asymmetric

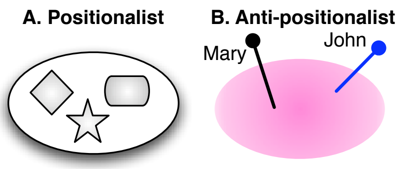
The 'positionalist' commitment

- Binary relation *killing* and identify the argument places—"argument positions" [Fine00] to have "distinguishability of the slots" [Cross02]—*killer* and *deceased* (loosely, a place for the nominative and a place for the accusative), assign *John* to *killer* and *the dragon* to *deceased* and order the three elements in any arrangement
- Relation(ship) and several distinguishable 'holes' and we put each object in its suitable hole.
- There are no asymmetrical relations, because a relationship R and its inverse R^{-}, or their instances, say, r and r', are *identical*, i.e., the same thing [Williamson85,Fine00,Cross02]

The 'positionalist' commitment

- Ingredients
  - (i) an n-ary relationship R with $A_1, \ldots, A_m$ participating object types ($m \leq n$),
  - (ii) n argument places $\pi_1, \ldots, \pi_n$, and
  - (iii) n assignments $\alpha_1, \ldots, \alpha_n$ that link each object $o_1, \ldots, o_n$ (each object instantiating an A_i) to an argument place ($\alpha \mapsto \pi \times o$)
- R, π_1, π_2, π_3, $r \in R$, $o_1 \in A_1$, $o_2 \in A_2$, $o_3 \in A_3$, then any of $\forall x, y, z(R(x, y, z) \rightarrow A_1(x) \wedge A_2(y) \wedge A_3(z))$ and its permutations with corresponding argument places—i.e., $R[\pi_1, \pi_2, \pi_3]$, and e.g., $R[\pi_2, \pi_1, \pi_3]$, and $[\pi_2\pi_3]R[\pi_1]$—all denote the same SoA under the same assignment o_1 to π_1, o_2 to π_2, and o_3 to π_3 for the extension
- Thus, $r(o_1, o_2, o_3)$, $r(o_2, o_1, o_3)$, and $o_2 o_3 r o_1$ are different representations of the same SoA where objects o_1, o_2, and o_3 are related to each other by means of relation r.

Graphical depictions



Problems with the 'positionalist' commitment

- From an ontological viewpoint, it requires identifiable argument positions to be part of the fundamental furniture of the universe.
- Practically, it requires something to finger-point to, i.e, to reify the argument places, and use it in the signature of the formal language, which is not clean and simple
- Symmetric relations, such as *adjacent to*, and relationships are problematic:
  - i. Take π_a and π_b of a symmetric binary relation r, assign o_1 to position π_a and o_2 to π_b in state s.
  - ii. One can do a reverse assignment of o_1 to position π_b and o_2 to π_a in state s'
  - iii. But then o_1 and o_2 do not occupy the same positions as they did in s, so s and s' must be different, which should not be the case.

The 'anti-positionalist' commitment

- No argument positions, but just a relation and objects that yield states by "combining" into "a single complex" [Fine00]
- Solves the problems with the standard view
- Solves the positionalist's problem with symmetric relations
- (How to formalise this idea in a KR language is another problem)

Example

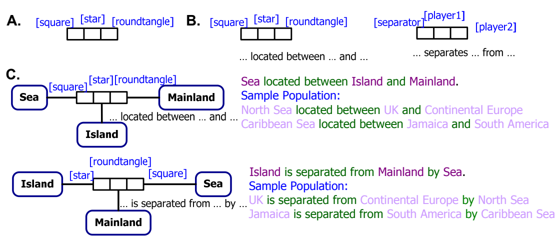
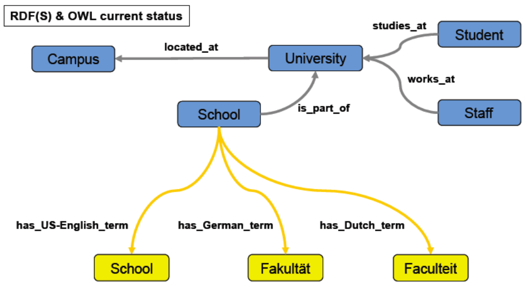


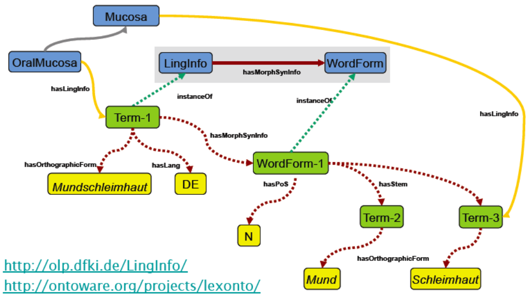
Figure: Positionalist examples in ORM. A: an ORM diagram rendering of Fig. 10-A; B: a reading added and a possible generalization of it, naming the relationship, e.g. *betweenness*; C: sample fact types and populations.

Ontologies in practice: Semantic Tagging—Classes, Terms



http://www.deri.ie/fileadmin/documents/teaching/tutorials/DERI-Tutorial-NLP.final.pdf

Ontologies in practice: Semantic Tagging—Lexicalized Ontologies



http://olp.dfki.de/LingInfo/
http://ontoware.org/projects/lexonto/

http://www.deri.ie/fileadmin/documents/teaching/tutorials/DERI-Tutorial-NLP.final.pdf

Examples (out of many)

- Generic tools: see `http://www.deri.ie/fileadmin/documents/teaching/tutorials/DERI-Tutorial-NLP.final.pdf` for a long list
- GoPubMed (Dietze et al, 2009)
  - Layer over PubMed, which indexes ± 19mln articles in the bio(medical) domain; pre-processing of the abstracts (advanced semantic tagging)
  - Results of the PubMed query are sorted according to terms in the ontology
- Question answer system AliQAn for agriculture (Vila and Ferrández, 2009)
  - Question assignment task too difficult for specialised domains
  - Add ontology to an open domain QA system, using AGROVOC and WordNet
- Attempto Controlled English (ACE), rabbit, etc.; grammar engine, template-based approach

Background

- Ontology development is time consuming
- Bottom-up ontology development strategies, of which one is to use NLP
- Where, if anywhere, can NLP make life easier for ontology development, and how?
- Current results are mostly discouraging, and depend on the approach, technique, and ontological commitment
  - We take a closer look at ontology learning limited to finding terms for a domain ontology

Bottom-up ontology development with NLP

- Usual parameters, such as purpose (in casu, document retrieval), formal language (an OWL species)
- A standard kind of ontology (not a comprehensive lexicalised ontology)
- Additional considerations for "text-mining ontologies"
  - Level of granularity of the terms to include (hypo/hypernyms)
  - How to deal with synonyms ('LDL I' and 'large LDL')
  - Handle term variations (e.g., 'LDL-I' and 'LDL I', 'Tangiers' disease' and 'Tangier's Disease')
  - Disambiguation; e.g. w.r.t. abbreviations

Method to test automated term recognition

- Compare the terms of a manually constructed ontology with the terms obtained from text mining a suitable corpus
- Build an ontology manually
  - Lipoprotein metabolism (LMO), 223 classes with 623 synonyms
- Create a corpus
  - 3066 review article abstract from PubMed, obtained with a 'lipoprotein metabolism' search
- Automatic Term Recognition (ATR) tools
  - Text2Onto: relative term frequency, TFIDF, entropy, hypernym structure of WordNet, Hearst patterns
  - Termine: statistics of candidate term, such as total frequency of occurrence, frequency of the term as part of other longer candidate terms, length of term
  - OntoLearn: linguistic processor and syntactic parser, Domain relevance and domain consensus
  - RelFreq: relative frequency of a term in a corpus
  - TFIDF: RelFreq + doc. frequency derived from all phrases in PubMed

Results

- OntoLearn excluded form analysis because it regenerated few terms
- Text2Onto only included in analysis for up to 300 abstracts (could not process all 3066)
- Precision for LMO 17-35% for top 50 terms, and 4-8% for top 1000 terms
- Precision for LMO + expert analysis of the automatically generated terms: up to 75% for top 50 terms, and up to 29% for top 1000 terms
- Termine good for the longer terms, RelFreq and TFIDF for the shorter terms

Results (cont'd)

Table 3: Coverage of LMO terminology in selected document sets. The table sets the upper limit of terms that can be found with text-mining: Even a large text base with 50,000 documents contains only 71% of LMO terms. TFIDF can predict up to 38% of LMO terms.

	LMO terminology predicted by TFIDF		*LMO terminology literally contained*
	1000	**all**	
300 review abstracts for "lipoprotein metabolism"	8.75%	15.35%	20.98%
3,066 abstracts for "lipoprotein metabolism"	14.99%	38.25%	53.00%
50,000 abstracts containing "lipoprotein"			71.22%

from Alexopoulou et al, 2008

What went wrong with some of the terms?

- LMO terms that were not in the 50k abstracts grouped into:
  - Rarely occurring terms: occur rarely even in the whole of PubMed
  - Rarely occurring variants of terms: e.g., 'free chol' (0, instead of 2622 for 'free cholesterol')
  - Very long terms; e.g, 'predominance of large low-density lipoprotein particles', which can be decomposed into smaller terms
  - Combinations of terms/variants; e.g., 'increased total chol' (0, instead of 116 for 'increased total cholesterol'),
  - Terms that should normally be easily found; e.g., 'diabetes type I' (126) and 'acetyl-coa c-acyltransferase', probably due to limited corpus
- Predicted terms, not in LMO: wrongly predicted ($\pm$25% of the TFIDF top50) or can be added to LMO ($\pm$40% of the TFIDF top50)

Typical NLP tasks

- Named Entity recognition/semantic tagging; e.g., "... the organisms were incubated at 37°C")
- Entity normalization; e.g., different strings refer to the same thing (full and abbreviated name, or single letter amino acid, three-letter aminoacid and full name: W, Trp, Tryptophan)
- Coreference resolution; in addition to synonyms (lactase and β-galactosidase), there as pronominal references (it, this)
- Grounding; the text string w.r.t. external source, like UniProt, that has the representation of the entity in reality
- Relation detection; *most of the important information in contained within the relations between entities*, NLP can be enhanced by considering semantically possible relations

Requirements for NLP ontologies

- Domain ontology (at least a taxonomy)
- Text model, concerns with classes such as *sentence*, *text position* and locations like *abstract*, *intorduction*
- Biological entities, i.e., contents for the ABox, often already available in biological databases on the Internet
- Lexical information for recognizing named entities; full names of entities, their synonyms, common variants and misspellings, and knowledge about naming, like *endo-* and *-ase*
- Database links to connect the lexical term to the entity represent in a particular database (the grounding step)
- Entity relations; represented in the domain ontology

MutationMiner use case

- See Witte et al. book chapter for details
- Ontology in OWL, in Protégé; with class name, textual definition and example instances
- Species info from the NCBI taxonomy; note the management of central *scientific name* and its synonyms, common variants and misspellings
- Uniprot and use of its back-links to the NCBI taxonomy

Discussion

- Significant upfront investments due to novelty and complexity of SWT
- Benefits:
  - Standardizes data exchange, consolidate disparate resources
  - Detecting inconsistencies (caused by, e.g. a pronoun with an incompatible relation to another textual entity)
- To do: Ontological NLP, enhancing standard NLP tools to take more of SWT into account

Overview

- Pure and applied life sciences use many diagrams
- Some diagram hand drawn, but more and more with software
- Come with their own 'icon vocabulary' and many diagrams
- Exploit such informal but structured representation of information to develop automatically (a preliminary version of) a domain ontology
- Formalize the 'icon vocabulary' in a suitable logic language, choose a foundational ontology (taxonomy, relations), categorise the formalised icons accordingly, load each diagram into the ontology, verify with the domain expert

Example of a PathwayAssist diagram

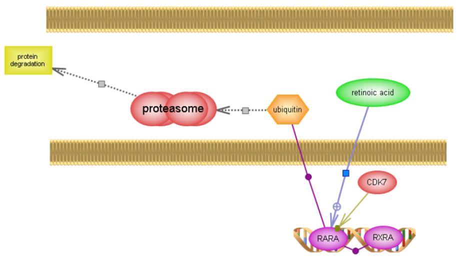
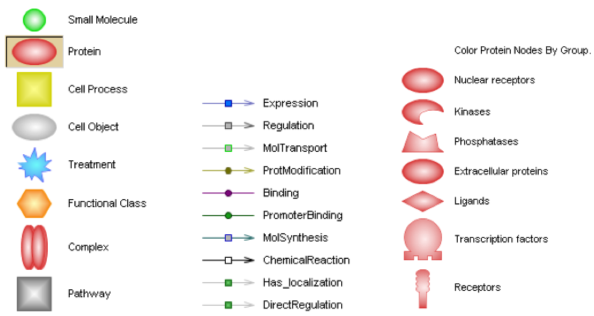


Figure: **Node** description: red: proteins, green: small molecules, orange: functional classes, yellow: cell processes, violet: nuclear receptors. **Link** description: grey dotted: regulation, violet solid: binding, yellow-green solid: protein modification, blue solid: expression.

Kindly provided by Kristina Hettne

PathwayAssist vocabulary

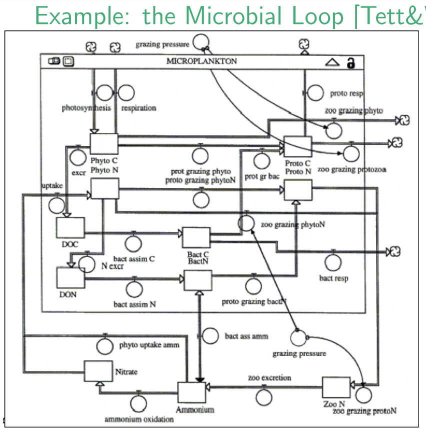


Kindly provided by Kristina Hettne

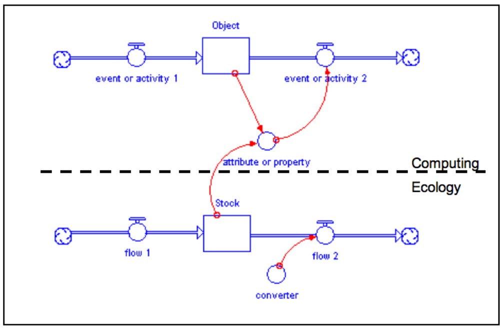
Case study motivation

- Experiment in 2005 (Keet, 2005), but progress made in ecology (Madin et al, 2008; MTSR'09 proceedings)
- Extensive use of modelling in ecology, but not much shared (depending on sub-discipline)
- Models used with independent software tools (DB and other applications)
- 'Legacy code' (procedural), moving toward more OO, and ontologies
- Requirement for (re-)analysis to upgrade legacy SW, develop new SW to meet increasing complexities and rising demands
- **use the opportunity to create a more durable, yet computationally usable, shared, agreed upon representation of the knowledge about reality**

Example: the Microbial Loop [Tett&Wilson04]



Key aspects in the ecological model: Flow, Stock, Converter, Action Connector



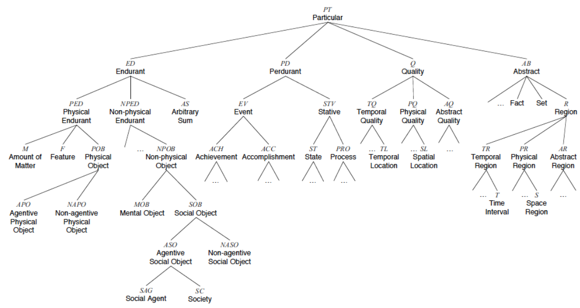
Informal 'Translation'

- A `Stock` correspond to a noun (particular or universal)
- `Flow` to verb
- `Converter` to attribute related to Flow or Stock
- `Action Connector` relates the former
- Object is candidate for an *Endurant*
- Event_or_activity for a method or *Perdurant*
- Converter maps to *Attribute_or_property*
- Action Connector candidate for *relationship* between any two of Flow, Stock and Converter

'Translation' w.r.t. DOLCE categories

- Basic mapping to DOLCE categories:
  - $\forall x((Stock(x) \leftrightarrow Entity(x)) \rightarrow ED(x))$
  - $\forall x((Flow(x) \leftrightarrow Entity(x)) \rightarrow PD(x))$
  - $\forall x((Converter(x) \leftrightarrow Entity(x)) \rightarrow (Q(x) \vee ST(x)))$
  - $\forall x(ActionConnector(x,y) \rightarrow Relationship(x,y))$

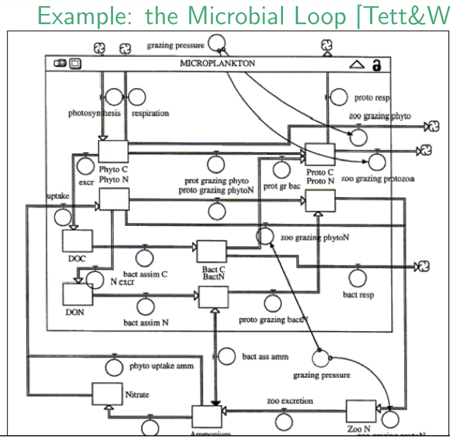
DOLCE categories



ML to Microbial Loop domain ontology

- Aim: to test translations with a real STELLA model
- ML's initial mapping to ontological categories contain 38 STELLA elements: 11 Stock/ED, 21 Flow/PD, 2 Converters/ST, 4 Action Connectors/Relationships
- The MicrobialLoop ontology has 59 classes and 10 properties
- Increase due to including DOLCE categories and implicit knowledge of ML that is explicit in MicrobialLoop

Example: the Microbial Loop [Tett&Wilson04]

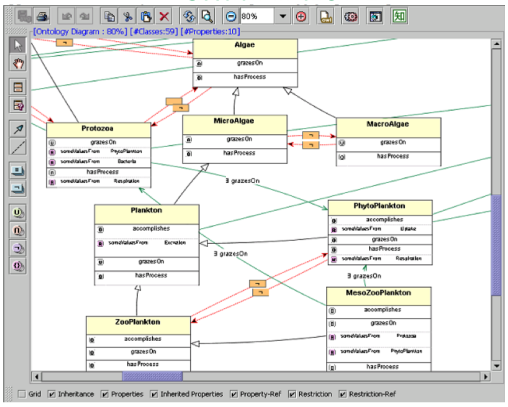


Section of more refined mapping to DOCLE categories

Phyto C	**NAPO**	**Phyto C = phytoplankton organic carbon. Phytoplankton is an APO, but 'phyto C' is *part* of the APO: only the organic carbon of the phytoplankton, not the organism as an active agent as such**
Phyto N	NAPO	Phyto N = phytoplankton nitrogen
DOC	NAPO	DOC = detrital organic carbon. Detritus is an ED with no unity, thus an amount of matter (M), but here, like with the organisms, there is focus on only a *part* of the NAPO
Nitrate	NAPO	Dissolved nitrate. Molecules are non agentive physical objects.
Flow		
Photosynthesis	PRO	To phytoplankton N
Respiration	PRO	From phytoplankton N
Prot gr bac	**PRO**	**Protozoa that are grazing on the Bacterial C**
Converter		
Grazing pressure	**ST**	**Acts on a PRO affecting the process of grazing; 'grazing pressure' is there (might reach zero), hence a ST.**
Action connector		
"1"	Yes	Acts on the mesozooplankton grazing on the protozoa, and acts on the mesozooplankton grazing on the phytoplankton: relation *hasGrazingPressure*

more mappings at http://www.meteck.org/supplDILS.html

Section in ezOWL



The serialized version of the ontology (section)

```
- <owl:Class rdf:ID="Protozoa">
    <owl:disjointWith rdf:resource="#Algae" />
    <owl:disjointWith rdf:resource="#Bacteria" />
  - <rdfs:subClassOf>
    - <owl:Restriction>
        <owl:onProperty rdf:resource="#hasProcess" />
        <owl:someValuesFrom rdf:resource="#Respiration" />
      </owl:Restriction>
    </rdfs:subClassOf>
  - <rdfs:subClassOf>
    - <owl:Restriction>
      - <owl:onProperty>
          <owl:ObjectProperty rdf:about="#grazesOn" />
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource="#PhytoPlankton" />
      </owl:Restriction>
    </rdfs:subClassOf>
  - <rdfs:subClassOf>
    - <owl:Restriction>
        <owl:someValuesFrom rdf:resource="#Bacteria" />
      - <owl:onProperty>
          <owl:ObjectProperty rdf:about="#grazesOn" />
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Microorganisms" />
  </owl:Class>
```


Discussion

- Formalising ecological natural, functional and integrative concepts
  - aids comparison of scientific theories
  - makes the implicit explicit, and more expressive than other modelling practices, therefore useful:
    - points to ambiguous sections,
    - part of/extra tool for doing science,
    - importance ontology maintenance, comparisons
- Modular, backbone or all-encompassing ontology/ies
- With the mappings, a quicker bottom-up development of ecological ontologies

To summarize

- Taxonomies insufficiently expressive compared to existing ecological modelling techniques
- Perspective of flow in ecological models cannot be represented adequately in a taxonomy
- More comprehensive semantics of formal ontologies
- Formalised mapping between STELLA and ontology elements facilitates bottom-up ontology development and has excellent potential for semi-automated ontology development
- STELLA as intermediate representation, widely used by ecologists and is translatable to a representation usable for ontologists

Overview

- Thesauri galore in medicine, education, agriculture, ...
- Core notions of **BT** broader term, **NT** narrower term, and **RT** related term (and auxiliary ones UF/USE)
- E.g. the Educational Resources Information Center thesaurus:
  ```
  reading ability
    BT ability
    RT reading
    RT perception
  ```
- E.g. AGROVOC of the FAO:
  ```
  milk
    NT cow milk
    NT milk fat
  ```
- *How to go from this to an ontology?*

Problems

- Lexicalisation of a conceptualisation
- Low ontological precision
- BT/NT is not the same as *is_a*, RT can be any type of relation: overloaded with (ambiguous) subject domain semantics
- Those relationships are used inconsistently
- Lacks basic categories alike those in DOLCE and BFO (ED, PD, SDC, etc.)

Simple Knowledge Organisation System(s): SKOS

- W3C standard intended for converting Thesauri, Classification Schemes, Taxonomies, Subject Headings etc into one interoperable syntax
  - Concept-based search instead of text-based search
  - Reuse each others concept definitions
  - Search across (institution) boundaries
  - Standard software
- Limitations:
  - 'unusual' concept schemes do not fit into SKOS (original structure too complex)
  - `skos:Concept` without clear properties (like in OWL) and still much subject domain semantics in the natural language text
  - 'semantic relations' have little semantics (`skos:narrower` does not guarantee it is *is_a* or *part_of*)

A rules-as-you-go approach

- A possible re-engineering procedure:
  - Define the ontology structure (top-level hierarchy/backbone)
  - Fill in values from one or more legacy Knowledge Organisation System to the extent possible (such as: which object properties?)
  - Edit manually using an ontology editor:
    - make existing information more precise
    - add new information
    - automation of discovered patterns (rules-as-you-go)

see (Soergel et al, 2004)

A rules-as-you-go approach

- A possible re-engineering procedure:
  - Define the ontology structure (top-level hierarchy/backbone)
  - Fill in values from one or more legacy Knowledge Organisation System to the extent possible (such as: which object properties?)
  - Edit manually using an ontology editor:
    - make existing information more precise
    - add new information
    - automation of discovered patterns (rules-as-you-go); e.g.
      - observation: *cow* NT *cow milk* should become *cow* <*hasComponent*> *cow milk*
      – pattern: *animal* <*hasComponent*> *milk* (or, more generally *animal* <*hasComponent*> *body part*)
      — derive automatically: *goat* NT *goat milk* should become *goat* <*hasComponent*> *goat milk*
      other pattern examples, e.g., *plant* <*growsIn*> *soil type* and *geographical entity* <*spatiallyIncludedIn*> *geographical entity*

Summary

Part V

Methods and methodologies

Outline

The landscape

- Difference between *method* and *methodology*
- Difference between writing down what you did (to make it a 'guideline') vs. experimentally validating a methodology
- Isn't ontology development just like conceptual data model development?
  - **yes**: e.g., interaction with the domain expert, data analysis
  - **no**: e.g., logic, automated reasoning, using (parts of) other ontologies, different scopes/purposes, specific isolated application scenario vs. general knowledge
- There are many methods for ontology development, but no up-to-date methodology

The landscape

- Multiple modelling issues in ontology development for the applied life sciences (e.g., part-of, uncertainty, prototypes, multilingual), methodological issues, highly specialised knowledge
- W3C's incubator group on modelling uncertainty, mushrooming of bio-ontologies, ontology design patterns, W3C standard OWL, etc.
- Solving the early-adopter issues moves the goal-posts
  - Which ontologies are reusable for one's own ontology?
  - What are the consequences choosing one ontology over the other?
  - The successor of OWL, draft OWL 2, has 5 languages: which one should be used for what and when?

Purposes

- Querying data by means of an ontology (OBDA) through linking databases to an ontology
- Database integration, (GO, OBO Foundry)
- Structured controlled vocabulary to link data(base) records and navigate across databases on the Internet ('linked data')
- Using it as part of scientific discourse and advancing research at a faster pace, (including experimental ontologies)
- Coordination among and integration of Web Services

Purpose

- Ontology in an ontology-driven information system destined for run-time usage, e.g., in scientific workflows, MASs, ontology-mediated data clustering, and user interaction in e-learning
- Ontologies for NLP, e.g.m annotating and querying Digital Libraries and scientific literature, QA systems, and materials for e-learning
- As full-fledged discipline "Ontology (Science)", where an ontology is a formal, logic-based, representation of a scientific theory
- Tutorial ontologies, e.g., the wine and pizza ontologies

Reusing ontologies

- Foundational ontologies
- Reference ontologies
- Domain ontologies that have an overlap with the new ontology;
- For each of them, resource usage considerations, such as
  - Availability of the resource (open, copyright)
  - If the source is being maintained or abandoned one-off effort;
  - Community effort, research group, and if it has already some adoption or usage;
  - Subject to standardization policies or stable releases;
  - If the ontology is available in the desired or required ontology language.

Example

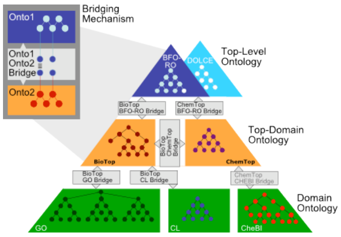


image from http://www.imbi.uni-freiburg.de/ontology/biotop/

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Languages – preliminary considerations

- Depending on the purpose(s) (and available resources), one ends up with either
  - (a) a large but simple ontology, i.e., mostly just a taxonomy without, or very few, properties (relations) linked to the concepts, where 'large' is, roughly, > 10000 concepts, so that a simple representation language suffices;
  - (b) a large and elaborate ontology, which includes rich usage of properties, defined concepts, and, roughly, requiring OWL-DL; or
  - (c) a small and very complex ontology, where 'small' is, roughly, $<$ 250 concepts, and requiring at least OWL 2 DL
- Certain choices for reusing ontologies or legacy material, or goal, may lock one a language
- $\Rightarrow$ Separate dimension that interferes with the previous parameters: the choice for a representation language

Languages

- Older KR languages (frames, obo, conceptual graphs, etc.)
- Web Ontology Languages:
  - OWL: OWL-Lite, OWL-DL, OWL full
  - OWL 2 with 4 languages to tailor the choice of ontology language to fit best with the usage scope in the context of a *scalable* and *multi-purpose* SW:
    - OWL 2 DL is most expressive and based on the DL language $\mathcal{SROIQ}$
    - OWL 2 EL fragment to achieve better performance with larger ontologies (e.g., for use with SNOMED-CT)
    - OWL 2 QL fragment to achieve better performance with ontologies linked to large amounts of data in secondary storage (databases); e.g. DIG-QuOnto
    - OWL 2 RL has special features to handle rules
- Extensions (probabilistic, fuzzy, temporal, etc.)
- Differences between expressiveness of the ontology languages and their trade-offs

Reasoning services

- Description logics-based reasoning services
  - The standard reasoning services for ontology usage: satisfiability and consistency checking, taxonomic classification, instance classification;
  - 'Non-standard' reasoning services to facilitate ontology development: explanation/justification, glass-box reasoning, pin-pointing errors, least-common subsumer;
  - Querying functionalities, such as epistemic and (unions of) conjunctive queries;
- Ontological reasoning services (OntoClean, RBox reasoning service)
- Other technologies (e.g., Bayesian networks)

		OWL Language					Ontology reuse		
	SKOS	2 QL	2 EL	2 DL	DL	Extensions	foundational	reference	domain
Purpose ⇓									
1. Query data	–	+	–	–	–	+	–	–	±
2. Database integration	+	+	+	–	–	±	±	±	+
3. Integration / record navigation	+	+	+	–	–	–	–	±	+
4. Part of scientific discourse	–	–	–	+	+	+	+	+	+
5. Web services orchestration	–	–	+	±	+	–	±	+	+
6. ODIS	±	+	+	–	±	±	±	+	+
7. ontoNLP	+	+	+	–	±	–	±	+	+
8. Science	–	–	–	+	±	+	+	+	–
9. Tutorial ontology	–	–	–	+	+	±	–	–	+
Reasoning services ⇓									
1. Standard	–	±	±	+	+	+			
2. Non-standard	–	±	±	+	+	–			
3. Querying	–	+	+	–	–	±			
4. Ontological	+	+	+	+	+	+			
Bottom-up ⇓									
1. Other KR/CM	–	±	±	+	+	–			
2. DB reverse	–	±	±	+	+	–			
3. Textbook models	–	–	±	+	+	+			
4. Thesauri	+	±	+	–	–	–			
5. Other semi-structured	±	±	+	–	–	–			
6. Text mining	+	±	+	–	–	–			
7. Terminologies	+	±	+	–	–	–			
8. Tagging	+	±	+	–	–	–			
Ontology reuse ⇓									
1. Foundational	–	–	–	+	±	–			
2. Reference	–	±	±	+	+	–			
3. Domain	±	±	+	+	+	–			

Table 1 Basic cross-matching between realistic combinations of parameters. The more complex dependencies, such as the interaction between purpose, language, and reasoning service, can be obtained from traversing the table (*purpose* ↔

OntoClean overview

- Problem: messy taxonomies on what subsumes what
- How to put them in the right order?
- OntoClean provides guidelines for this (see to Guarino & Welty, 2004 for an extended example)
- Based on philosophical principles, such as identity and rigidity (see Guarino & Welty's EKAW'00 and ECAI'00 papers for more information on the basics)

Basics

- A property of an entity is *essential* to that entity if it must be true of it in every possible world, i.e. if it necessarily holds for that entity.
- Special form of essentiality is *rigidity*

Definition (+R)

A *rigid* property ϕ is a property that is essential to *all* its instances, i.e., $\forall x \phi(x) \rightarrow \Box \phi(x)$.

Definition (-R)

A *non-rigid* property ϕ is a property that is not essential to *some* of its instances, i.e., $\exists x \phi(x) \wedge \neg \Box \phi(x)$.

Basics

Definition (~R)

An *anti-rigid* property ϕ is a property that is not essential to *all* its instances, i.e., $\forall x \phi(x) \rightarrow \neg \Box \phi(x)$.

Definition (¬R)

A *semi-rigid* property ϕ is a property that is non-rigid but not anti-rigid.

- Anti-rigid properties cannot subsume rigid properties

Basics

- *Identity*: being able to recognize individual entities in the world as being the same (or different)
- *Unity*: being able to recognize all the parts that form an individual entity; e.g., ocean carries unity (+U), legal agent carries no unity (-U), and amount of water carries anti-unity ("not necessarily wholes", ~U)
- *Identity criteria* are the criteria we use to answer questions like, "is that my dog?"
- Identity criteria are conditions used to determine equality (sufficient conditions) and that are entailed by equality (necessary conditions)

Basics

Definition

A non-rigid property carries an IC Γ iff it is subsumed by a rigid property carrying Γ.

Definition

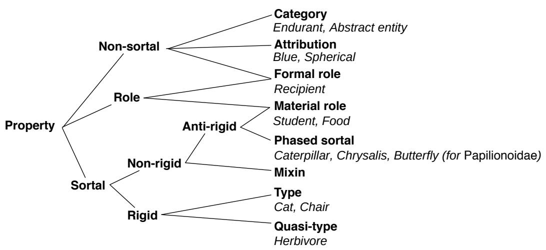
A property ϕ supplies an IC Γ iff i) it is rigid; ii) it carries Γ; and iii) Γ is not carried by all the properties subsuming ϕ. This means that, if ϕ inherits different (but compatible) ICs from multiple properties, it still counts as supplying an IC.

- Any property carrying an IC: +I (-I otherwise).
- Any property supplying an IC: +O (-O otherwise); "O" is a mnemonic for "own identity"
- +O implies +I and +R

Formal ontological property classifications

+O	+I	+R	+D	Type	Sortal
			-D		
-O	+I	+R	+D	Quasi-Type	
			-D		
-O	+I	~R	+D	Material role	
-O	+I	~R	-D	Phased sortal	
-O	+I	¬R	+D	Mixin	
			-D		
-O	-I	+R	+D	Category	Non-Sortal
			-D		
-O	-I	~R	+D	Formal role	
-O	-I	~R	-D	Attribution	
		¬R	+D		
			-D		

Formal ontological property classifications



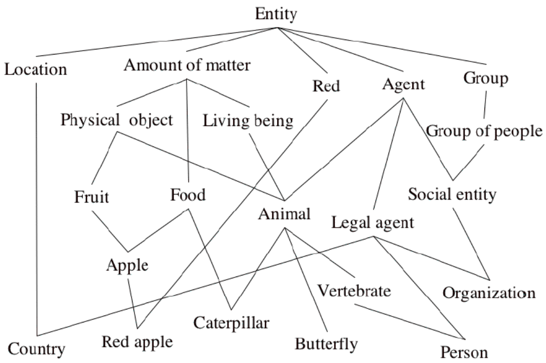
Basic rules

- Given two properties, p and q, when q subsumes p the following constraints hold:
  1. If q is anti-rigid, then p must be anti-rigid
  2. If q carries an IC, then p must carry the same IC
  3. If q carries a UC, then p must carry the same UC
  4. If q has anti-unity, then p must also have anti-unity

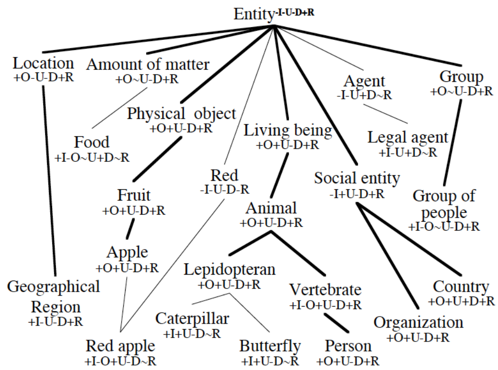
5. Incompatible IC's are disjoint, and Incompatible UC's are disjoint

- And, in shorthand:
  6. $+R \not\subset \sim R$
  7. $-I \not\subset +I$
  8. $-U \not\subset +U$
  9. $+U \not\subset \sim U$
  10. $-D \not\subset +D$

Example: before



Example: after



Overview

- Domain experts are expert in their subject domain, which is not logic
- Modellers often do not understand the subject domain well
- The more expressive the language, the easier it is to make errors or bump into unintended entailments
- Simple languages can represent more than it initially may seem (by some more elaborate encoding), which clutters the ontology and affects comprehension
- In short: people make errors (w.r.t. their intentions) in the modelling task, and automated reasoners can help fix that

Overview

- Using automated reasoners for 'debugging' ontologies, requires one to know about reasoning services
- Using standard reasoning services
- New reasoning services tailored to pinpointing the errors and explaining the entailments

Common errors

- Unsatisfiable classes
  - In the tools: the unsatisfiable classes end up as direct subclass of `owl:Nothing`
  - Sometimes one little error generates a whole cascade of unsatisfiable classes
- Satisfiability checking can cause rearrangement of the class tree and any inferred relationships to be associated with a class definition: 'desirable' vs. 'undesireable' inferred subsumptions
- Inconsistent ontologies: all classes *taken together* unsatisfiable

Common errors

- Basic set of clashes for concepts (w.r.t. tableaux algorithms) are:
  - Atomic: An individual belongs to a class and its complement
  - Cardinality: An individual has a max cardinality restriction but is related to more distinct individuals
  - Datatype: A literal value violates the (global or local) range restrictions on a datatype property
- Basic set of clashes for KBs (ontology + instances) are:
  - Inconsistency of assertions about individuals, e.g., an individual is asserted to belong to disjoint classes or has a cardinality restriction but related to more individuals
  - Individuals related to unsatisfiable classes
  - Defects in class axioms involving nominals (`owl:oneOf`, if present in the language)

Where are we?

- Parameters that affect ontology development, such as purpose, base material, language
- Methods, such as reverse engineering text mining to start, OntoClean to improve
- Tools to model, to reason, to debug, to integrate, to link to data
- Methodologies that are coarse-grained: they do not (yet) contain all the permutations at each step, i.e. *what* and *how* to do each step, given the recent developments;
- e.g. step x is "knowledge acquisition", but what are it component-steps?

Example methodology: METHONTOLOGY

- Basic methodology:
  - specification: why, what are its intended uses, who are the prospective users
  - conceptualization, with intermediate representations
  - formalization (transforms the domain-expert understandable 'conceptual model' into a formal or semi-computable model)
  - implementation (represent it in an ontology language)
  - maintenance (corrections, updates, etc)
- Additional tasks (as identified by METHONTOLOGY)
  - Management activities (schedule, control, and quality assurance)
  - Support activities (knowledge acquisition, integration, evaluation, documentation, and configuration management)
- Applied to chemical, legal domain, and others (More comprehensive assessment of extant methodologies in Corcho et al, 2003)

MOdelling wiKI

- MoKi is based on a **SemanticWiki**, which is used for collaborative and cooperative ontology development
- It enables actors with different expertise to develop an "enterprise model"[9]: use both *structural (formal) descriptions* and *more informal* and *semi-formal* descriptions of knowledge
- ⇒ access to the enterprise model **at different levels of formality**: informal, semi-formal and formal
- more info and demo at `http://moki.fbk.eu`

[9] enterprise model: "a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise"

Extending the methodologies

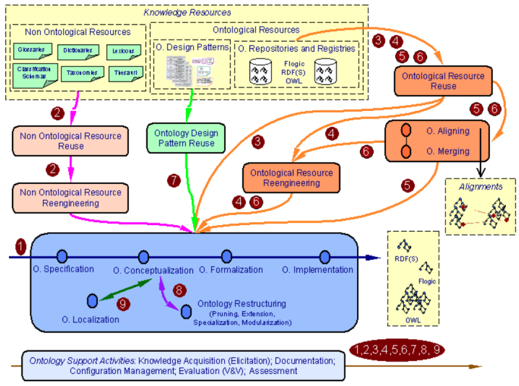
- METHONTOLOGY, MoKi, and others (e.g., On-To-Knowledge, KACTUS approach) are for developing one *single* ontology
- Changing landscape in ontology development towards building "ontology networks"
- Characteristics: dynamics, context, collaborative, distributed
- E.g., the emerging NeOn methodology

Extending the methodologies: NeOn

- NeOn's "Glossary of Activities" identifies and defines 55 activities when ontology networks are collaboratively built
- Among others: ontology localization, -alignment, -formalization, -diagnosis, -enrichment etc.
- Divided into a matrix with "required" and "if applicable"
- Embedded into a comprehensive methodology (under development)

(more info in neon_2008_d5.4.1.pdf)

Scenarios for Building Ontology Networks



Tools

- Thus far, no tool gives you everything
  - WebODE to support METHONTOLOGY with a software application
  - Protégé with its plugins. a.o.: ontology visualisation, querying, OBDA, etc.
  - NeOn toolkit aims to be a "open source multi-platform ontology engineering environment, which aims to provide comprehensive support for all activities in the ontology engineering life-cycle"; 45 plugins
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - KAON, SWOOP, etc.
  - Specialised tools for specific task, such as ontology integration and evaluation (e.g. Protégé-PROMPT, ODEClean)
  - RDF-based ones, such as Sesame
- *Longer list and links to more lists of tools in the accompanying text and references*

Summary

Part VI

Extra topics

Outline

SWT challenges or failures?

- Challenge: solution to problem y not possible yet (or very difficult to achieve) with current SWT, but in theory is (expected to be) feasible
- Failure: technology x claims to solve problem y but it does not and will not do so, or technology x is developed for a non-existing problem but does not solve real problems
  - Is y one that, at least in theory, can be solved with SWT?
  - Was y described too broadly, so that it solves only a subset of the cases?
  - Were there perhaps additional requirements put on a solution?
- Are disconnected technologies with ad-hoc patches a challenge to solve or a failure in devising a generic suite?
- A failure according to one may be considered a challenge by another
- Offer and demand, perceptions, perspectives, expectations

A few general issues

- RDF triple stores vs. RDBMSs vs OWL ABoxes in memory; more generally:
  - Making 'legacy' (operational) systems 'Semantic Web compliant'
  - Add a 'wrapper' over the legacy system so that from the outside it looks like it uses SWT
- How to integrate rules other than at instance level
- Modularization
- Semantics-based language transformations
- Coordination among tools with different functionalities

Language limitations considerations

- Known trade-offs between expressiveness and computational complexity
- Different ontology developers and their scopes (and purposes of the ontologies):
  - to some, there is more in OWL/OWL2 than needed and used
  - to some, there is not enough
- From a logician's perspective, language limitations are not failures per sé, only *challenges* to find the more interesting and useful combinations of features
- From a modeller's perspective, the trade-offs can be such that it is deemed a *failure* with respect to the expectations and application needs

Which language do we need?

- The (reflexive, antisymmetric, transitive) parthood relation
- Each Government has as members at least 10 Ministers
- A father is necessarily male
- Each plane passenger boards the aircraft after having checked in
- Swedish people are very tall
- The class of people who are young
- Generally, birds do fly
- 90% of the Italians have brown eyes
- Any two people are related to each other in one way or another

Limitations as identified by users/modellers (a.o., Schulz et al, 2009)

- *n*-ary relations, where $n > 2$
- "Hepatitis hasSymptom Fever in most but not all cases"
  - What about doing it with probabilistic default knowledge?
  - $(\psi \mid \phi)[l, u]$ as "generally, if an object belongs to ϕ, then it belongs to ψ with a probability in $[l, u]$"
  - e.g., $(\exists \texttt{hasSymptom.Fever} \mid \texttt{Hepatitis})[1, 1]$
- "In 2000, worldwide prevalence of diabetes mellitus was 2.8%"
  - Probabilistic, or arithmetic, or what have we?
  - First, it assumes some class `Human` and a class `HumanDiabetesMellitus`, where some of the instances of the former have (are `bearerOf`) an instance of the latter
  - Second, we have some notion of `prevalence`, but what is it associated to (a property of)? of the human *population* in the world, not a property of an individual human

Limitations as identified by users/modellers (Schulz et al, 2009)

- ... Diabetes example continued
  - Authors' proposal to put it in the ABox with arithmetic operators, e.g. "$\frac{|DiabeticHuman|}{|Human|} = 0.028$"
  - Another option: put in TBox with a data property, e.g., $\texttt{HumanDiabetesMellitus} \sqsubseteq \exists \texttt{hasPrevalence.real}$
  - Yet another: represent the *probability* of a human having diabetes mellitus
  - What are the pros and cons of each option w.r.t. subject domain semantics, Ontology, and the ontology languages?
- Problems with Drug Abuse Prevention (in SNOMED CT)
  - $\texttt{DrugAbusePrevention} \sqsubseteq \texttt{Procedure} \sqcap \exists \texttt{hasFocus.DrugAbuse}$
  - $\texttt{DrugAbusePrevention} \equiv \texttt{Procedure} \sqcap \exists \texttt{hasParticipant.Person} \sqcap \exists \texttt{causes.(State} \sqcap \texttt{hasParticipant.(Person} \sqcap \exists \texttt{participatesIn.}\neg \texttt{DrugAbuse))}$

Limitations as identified by users/modellers (Schulz et al, 2009)

- "Concussion of the brain without loss of consciousness", and the temporal aspects
- "aspirin prevents myocardial infarction"
  - Let us assume that is total prevention (though we could add a probability to it)
  - This only holds for humans actually ingesting aspirin, not for the substance itself
  - It then intends to say that the human taking aspirin will not have a myocardial infarction *at all times in the future*, which can be represented in a suitable temporal logic with the $\Box^+$
  - e.g., `AspirinIntake` $\sqsubseteq$ $\Box^+$`prevents.MyocardialInfarction`, or `MyocardialInfarction` $\sqsubseteq$ $\Box^+$`preventedBy.AspirinIntake`, or `AspirinIntake` $\sqsubseteq$ $\Box^+$`hasPhysiologicalEffect.`$\neg$`MyocardialInfarction` ?

Introduction on reasoning scenarios

- The standard reasoning services are obviously sorted out
- Performance issues for the 'debugging' and explanation reasoning, and how to provide the 'best' explanation
- Querying OWL 2 DL, and any ABox data
- Additional reasoning scenarios with 'standard' ontologies
- Reasoning over fuzzy, rough, probabilistic, possibilistic, time, ... ontologies and data

Scenarios

1. Supporting the ontology development process
2. Classification
3. Model checking (violation)
4. Finding gaps in an ontology & discovering new relations
   - Deriving types and relations from instance-level data
   - Computing derived relations at the type level
5. Comparison of two ontologies ([logical] theories)
6. Reasoning with part-whole relations
7. Using (including finding inconsistencies in) a hierarchy of relations
8. Reasoning across linked ontologies
9. Complex queries

Checking against instances

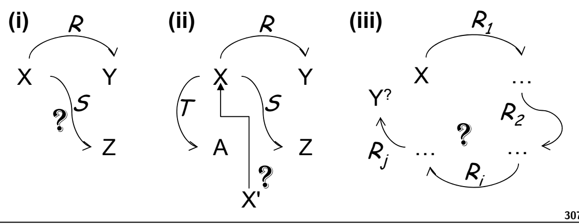
- Usual model checking
- Model checking against *real* instances in the ABox/Database
  - For each DL-concept in the OWL-formalised ontology (representing a universal), there has to be at least one ABox instance (as representation of the entity in reality)
  - To spot "redundant" DL-concepts w.r.t. the data-needs
- Model violation
  - Reducing the amount of instances to only those that do not violate the TBox (or: the more inconsistencies, the better)
  - For instance, to find a few candidate molecules that satisfy a given set of properties, out of a large pool of possibly suitable molecules; e.g., for drug discovery in pharmainformatics, tyre production

Discovering information

- The idea is that the combination of bio-ontologies, instances, and automated reasoning services somehow can find either the missing relations, or the types, or both
- How can one find what is, or may, not be in the ontology but ought to be there?
- At the TBox-level
  - computing derived relations (object properties)
  - find out where relations that are known by the developer have not yet been added to the ontology (finding 'known gaps')
  - add 'ontological' notions with top type 'whole' in a partonomy; e.g., 17 types of macrophage in the FMA each must be part of something
  - flag classes that have no relation (no or no is_a) to anything else in the ontology

Discovering information

- For the TBox through querying the data (ABox, RDBMS)
  - i. "for each x:X, y:Y, r:R, XRY, does there exist a z:Z, s:S, such that there exist ≥ 1 x and xsz?"
  - ii. "for each x:X, y:Y, r:R, XRY, does there exist an xsz and an xta where z:Z, s:S, a:A, t:T hold?"
  - iii. Find-me-anything-you-have: "for each x:X, return any $r_1, \ldots r_n$, their type of role and the concepts $Y_1, \ldots Y_n$ they are related to"



Building ontologies involves humans

- Building an ontology is, generally, an *interdisciplinary* (transdisciplinary?) endeavour
- Different disciplines with different mores, goals
- The collaboration requires patience, respect, capability to listen, compromise
- More slides in a separate file, time permitting