Housekeeping

- This course consists of lectures, exercises, and a mini-project that will be carried out in small groups.
- Each lecture takes about 2 hours, labs and mini-project another 1-2 hours.
- 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.
- There is a more comprehensive list of topics that can be spread out over an entire MSc degree programme, and other courses will be given later that will address some topics in more detail.

Outline

- Introduction to ontologies
- Ontology Languages: OWL and OWL2
- Foundational and top-down aspects of ontology engineering
- Bottom-up ontology development
- Methods and methodologies
- Extra topics

Part I

Introduction to ontologies

Background

- Aristotle and colleagues: Ontology
- Engineering: ontologies (count noun)
- Investigating reality, representing it
- Putting an engineering artifact to use

What then, is this engineering artifact?
What is an ontology? What is the usefulness of an ontology? Success stories

A few definitions

- Most quoted: “An ontology is a specification of a conceptualization” (by Tom Gruber, 1993)
- 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)
- “with an ontology being equivalent to a Description Logic knowledge base” (Horrocks et al, 2003)

Description Logic knowledge base

Ontology

TBox (with intensional knowledge)

ABox (with extensional knowledge involving objects and values)

Ontologies and meaning (Guarino, 2002)

Ontologies and reality

Reality

Ontology

Language L

Ontological commitment K

Tarksi\textquotesingle s interpretation I

Intended models \( I_\mathcal{L}(L) \)

Ontology models \( I_\mathcal{L}(L) \)

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.” ⇒ 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)
What is an ontology? What is the usefulness of an ontology? Success stories

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

And graphically

Examples in different application areas, using different features

- Data integration
- Instance classification
- 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 an 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?
**What is an ontology?**

What is the usefulness of an ontology?

Success stories

**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 and others came on board (see [http://www.geneontology.org](http://www.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

**GO contents example (process)**

![GO contents example (process)](image)

**GO contents example (cellular component)**

![GO contents example (cellular component)](image)

**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 its 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 Open Biological Ontologies principles...
  - open,
  - orthogonal,
  - same syntax,
  - continuants, and occurrents
- ... to one for the Open Biological and Biomedical Ontologies:
  - developed in a collaborative effort
  - usage of common relations that are unambiguously defined (in case: 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' ...")

- Aimed at "coordinated evolution of ontologies to support biomedical data integration"

**OBO Foundry coverage (canonical ontologies)**

**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)

**How it can be done**

- Develop ontology for the subject domain, in OWL
  - Extract knowledge from peer-reviewed literature
  - Protein phosphatases; e.g., Class NPhosphatase Complete
    - (hasDomain two TyrosinePhosphataseCatalyticDomain) and
    - (hasDomain some TransmembraneDomain) and
     - (hasDomain some CarbonicAnhydraseDomain)
- Obtain instance data
  - Process protein sequences by InterProScan
  - Transform into OWL
  - Put it together in some system with a reasoner
  - InstanceStore
  - Racer reasoner

**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
What is an ontology? What is the usefulness of an ontology? Success stories

Results

- Human phosphatases:
  - The reasoner as good as human expert classification
  - Identification of additional p-domains, refined the classification into further subtypes
- A. fumigatus phosphatases:
  - 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

Web-based, graphical, ontology-based querying of lots of data (Calvanese et al, 2010)

- The setting:
  - Large amounts of data available on the Web, which can be accessed by canned or precomputed queries presented via web forms, or SQL
  - Domain expert wants more flexibility in data analysis and hypothesis testing, and independence from the sysadmin to do the queries for them
- The problems:
  - There is no proper, real life, use case that demonstrates the benefits of scalable, user-useable, Ontology-Based Data Access
  - That one has to know how the data is stored, instead of concerning oneself with what kind of information is in the database
  - Domain expert-unfriendly query mechanisms (SQL, SPARQL)

Idea

- Ontology-Based Data Access, to achieve data access at the ‘what-layer’, i.e., adding a semantic layer to the database
- Web-based, like most other bioinformatics resources
- Graphical querying to make it usable by the domain expert
- Usage of, mainly, reasoning services for querying the ontology and the data

How it can be done

- Develop ontology of the subject domain, in OWL
- Reverse engineering existing database HGT-DB
  - (http://genomes.urv.cat/HGT-DB/), further manual improvements to create a proper conceptual data model
- Simplify this conceptual data model into the appropriate OWL language (DL-Lite, which is roughly OWL 2 QL)
- Create mappings between the terms in the ontology to SQL queries over the database
  - Using the OBDA Plugin for Prot´eg´e
  - Oracle database (can also be PostgreSQL, DB2, ...), 4GB genomics database (HGT-DB), tables with 16-46 columns
- Connect this to an OBDA-enabled reasoner
  - In this case: QuOnto (but can be others)
What is an ontology? What is the usefulness of an ontology? Success stories

Architecture

Example: Diagram – DL-lite_A correspondence

Formalisation of the graphical elements

Example: mapping concepts & relations of the Ontology to SQL query over the relational database

Queries

- SPARQL queries for conjunctions and equalities
- Epistemic queries in EQL-Lite for constraints involving inequalities and string matching
  - Imposes constraints on top of the certain answers retrieved by a DL-Lite conjunctive query
  - Result obtained by:
    1. computing the certain answers for the CQ $q(y) \rightarrow \text{conj}(\vec{x})$ (with $\text{conj}(\vec{x})$ the conjunction of atoms, and $y$ a vector comprising the variables in $\vec{x}$ and in $\vec{w}$),
    2. filtering the resulting tuples according to the constraint expression $\text{cons}(\vec{x}, \vec{y})$, and
    3. projecting onto $\vec{x}$ (a vector comprising the variables corresponding to the highlighted nodes in the query pane)

Results

- Demo of the Wonder system (Web-ONtology baseD Extraction of Relational data)
  - Builds upon the theory, technology, and implementation developed for Ontology-Based Data Access
  - Graphical ontology browsing, query formulation, and query execution in a Web browser
  - Rigorous formal characterisation and uses a coupling with an OWL file
  - EQL-Qs (in SPARQL syntax) and EQL-Lite queries managed by the DIG-QuOnto reasoner
  - Performance good, GUI insignificant influence on performance
  - Usability testing: usable, and domain experts came up with a range of new queries to analyse the data
What is an ontology? What is the usefulness of an ontology? Success stories

Additional features

- **Wonder** currently focuses on querying one database
- OBDA architecture allows for querying incomplete data (data integration scenario)²
- Querying of the application ontology itself, as well as a combination of querying the ontology and the data³
  - in certain settings, possible to include queries that use the knowledge in the ontology for which there is no data in the database, and still retrieve the right results


Sample query in OBDA Plugin

q(x) :- Device(x), assistsWith(x, y), UpperLimbMobility(y)

Informal overview of kind of knowledge in ADOLENA

Summary

What is an ontology?

What is the usefulness of an ontology?

Success stories

The GO and data integration
Exploiting the classification reasoning services
Scalable querying of ontologies and data

Outline

Limitations of RDFS

OWL

Design of OWL
OWL and Description Logics
OWL Syntaxes

Limitations

OWL 2

OWL 2 DL

OWL 2 profiles

OWL 2 EL

OWL 2 QL

OWL 2 RL

Reasoning

Part II

Ontology Languages: OWL and OWL2
Toward one ontology language

- Plethora of ontology languages, lack of a lingua franca (hence, onto 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)

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
- 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
- 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/optimal 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
  - (in)quality
  - (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
Limitations of RDFS

OWL Limitations

OWL 2

OWL 2 profiles

Reasoning

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++)

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

OWL Full entails:

friend rdf:type owl:ObjectProperty .

John rdf:type owl:Thing . 
friend rdf:domain owl:Thing .
friend rdf:range owl:Thing .

John rdf:type owl:Thing . 
friend rdf:domain owl:Thing .
friend rdf:range owl:Thing .

John rdf:type owl:Thing . 
friend rdf:domain owl:Thing .
friend rdf:range owl:Thing .

John rdf:type owl:Thing . 
friend rdf:domain owl:Thing .
friend rdf:range owl:Thing .

More on OWL and Description Logics

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

<table>
<thead>
<tr>
<th>OWL Construct</th>
<th>DL</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>Human $\sqcup$ Male</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \cup \ldots \cup C_n$</td>
<td>$\cup$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$C$</td>
<td>$\neg C$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${a_1, \ldots, a_n}$</td>
<td>(john, mary)</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P. C$</td>
<td>$\forall C. \forall P. C$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P. C$</td>
<td>$\exists C. \exists P. C$</td>
</tr>
<tr>
<td>value</td>
<td>$\geq n_P. C$</td>
<td>$\geq n_P$.</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq n_P. C$</td>
<td>$\geq n_P$.</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq n_P. C$</td>
<td>$\leq n_P$.</td>
</tr>
<tr>
<td>cardinality</td>
<td>$= n_P. C$</td>
<td>$= n_P$.</td>
</tr>
</tbody>
</table>

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

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#)
- OWL gathers information into ontologies stored as documents written in RDF/XML, things like ol: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

Example from [OwlGuide]:

```xml
<owl:Class rdf:ID="Wine" />
<owl:Class rdf:ID="Pasta" />
<owl:Class rdf:ID="Fruit" />
<owl:Class rdf:ID="Document" />
<owl:Class rdf:ID="Food" />
```

OWL in RDF/XML

```
<owl:Class rdf:ID="Wine" />
<owl:Class rdf:ID="Pasta" />
<owl:Class rdf:ID="Fruit" />
<owl:Class rdf:ID="Document" />
<owl:Class rdf:ID="Food" />
```

OWL constructs

<table>
<thead>
<tr>
<th>OWL Axiom</th>
<th>DL</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>$\sqsubseteq$</td>
</tr>
<tr>
<td>EquivalentClasses</td>
<td>$C_1 \equiv C_2$</td>
<td>$\equiv$</td>
</tr>
<tr>
<td>SubPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>$\sqsubseteq$</td>
</tr>
<tr>
<td>EquivalentProperties</td>
<td>$P_1 \equiv P_2$</td>
<td>$\equiv$</td>
</tr>
<tr>
<td>SameIndividual</td>
<td>$\forall i. C_i$</td>
<td>$\forall C_i$.</td>
</tr>
<tr>
<td>DisjointClasses</td>
<td>$C_i \sqcap C_j$</td>
<td>$\sqcap$</td>
</tr>
<tr>
<td>Different Individuals</td>
<td>$P_i \neq P_j$</td>
<td>$\neq$</td>
</tr>
<tr>
<td>inverseOf</td>
<td>$\neg P$</td>
<td>$\neg P$.</td>
</tr>
<tr>
<td>Transitive</td>
<td>$P^+ \sqsubseteq P$</td>
<td>$\sqsubseteq$</td>
</tr>
<tr>
<td>Symmetric</td>
<td>$P \equiv P^-$</td>
<td>$\equiv$</td>
</tr>
</tbody>
</table>

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-readable ones
  - e.g., Manchester syntax, informal and limited matching with UML

```
<owl:Class rdf:ID="Wine" />
<owl:Class rdf:ID="Pasta" />
<owl:Class rdf:ID="Fruit" />
<owl:Class rdf:ID="Document" />
<owl:Class rdf:ID="Food" />
```

OWL Abstract syntax

```
Class( professor partial ) Class( associateProfessor partial academicStaffMember )
```

```
DisjointClasses ( associateProfessor assistantProfessor )
DisjointClasses ( professor associateProfessor )
```

```
Class( faculty complete academicStaffMember )
```

```
In DL syntax:
associateProfessor ¬ academicStaffMember
associateProfessor ¬ assistantProfessor
professor ¬ associateProfessor
```

```
faculty = 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))

Individual (940352 type(academicStaffMember) value(age "39" &xsd;integer))
ObjectProperty(isTaughtBy Functional)
Individual (CIT1111 type(course) value(isTaughtBy 940352) value(isTaughtBy 940318))

DifferentIndividuals (940318 940352) DifferentIndividuals (940352 940111 940318)

Class(firstYearCourse partial restriction (isTaughtBy allValuesFrom (Professor)))
Class(mathCourse partial restriction (isTaughtBy hasValue (940352)))
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)))

In DL syntax:

In DL syntax:

⊤⊑∀ ag.xsd :nonNegativeInteger
⊤⊑∀ isTaughtBy −.course
⊤⊑∀ isTaughtBy. academicStaffMember
isTaughtBy ⊑involves teaches = isTaughtBy
⊤⊑∀ teachesacademicStaffMember
⊤⊑∀ teaches course
lecturesIn ⊑teaches
hasSameGradeAs ⊓hasSameGradeAs
hasSameGradeAs ⊓hasSameGradeAs−
⊤⊑∀ hasSameGradeAs−.student
⊤⊑∀ hasSameGradeAs.student

In DL syntax:

firstYearCourse ⊑∀isTaughtBy.Professor
mathCourse ⊑∀isTaughtBy. (940352)
academicStaffMember ⊑∀teaches.undergraduateCourse
course ⊑∀isTaughtBy
department ⊑∀10hasMember ⊓30hasMember
More examples

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:

\[
\text{course } \sqsubseteq \text{staffMember} \\
\text{peopleAtUni } \sqsubseteq \text{staffMember } \sqcap \text{student} \\
\text{facultyInCS } \sqsubseteq \text{faculty } \sqcap \{\text{belongsTo (CSDepartment)}\} \\
\text{adminStaff } \sqsubseteq \text{staffMember } \sqcap \neg(\text{faculty } \sqcup \text{techSupportStaff})
\]

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 \( G \models_{\text{RDFS}} G' \) then \( G \models_{L} G' \), and ideally
if \( G \models_{L} G' \) then \( G \models_{\text{RDFS}} G' \)

Syntactically layering OWL on RDF(S)

<table>
<thead>
<tr>
<th>OWL Lite, OWL DL</th>
<th>OWL Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>- OWL Lite, OWL DL subsets of RDF</td>
<td></td>
</tr>
<tr>
<td>- Allowed triples defined through mapping from abstract syntax</td>
<td></td>
</tr>
<tr>
<td>- Partial layering:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- every OWL Lite/DL ontology is an RDF graph</td>
</tr>
<tr>
<td></td>
<td>- some RDF graphs are OWL Lite/DL ontologies</td>
</tr>
<tr>
<td>- OWL Lite/DL vs. RDF</td>
<td></td>
</tr>
<tr>
<td>- RDF Graph defined through translation from Abstract Syntax</td>
<td></td>
</tr>
<tr>
<td>- Example:</td>
<td></td>
</tr>
<tr>
<td>- Class(Human partial Animal restriction(hasLegs cardinality(2)) restriction(hasName allValuesFrom(xsd:string)))</td>
<td></td>
</tr>
</tbody>
</table>

Semantically layering OWL on RDF(S)

<table>
<thead>
<tr>
<th>OWL Lite, OWL DL</th>
<th>OWL Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>- OWL Full encompasses RDF</td>
<td></td>
</tr>
<tr>
<td>- Complete layering:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- every OWL Full is an RDF graph</td>
</tr>
<tr>
<td></td>
<td>- all RDF graphs are OWL Full ontologies</td>
</tr>
<tr>
<td>- OWL Full vs. RDF</td>
<td></td>
</tr>
<tr>
<td>- Not every RDF graph is OWL Lite/DL ontology</td>
<td></td>
</tr>
<tr>
<td>- Example:</td>
<td></td>
</tr>
<tr>
<td>- A rdf:type A</td>
<td></td>
</tr>
<tr>
<td>- How to check whether an RDF graph ( G ) is OWL DL?</td>
<td></td>
</tr>
<tr>
<td>- Construct an OWL ontology ( O ) in Abstract Syntax</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Translate to RDF graph ( G' )</td>
</tr>
<tr>
<td></td>
<td>- If ( G'=G ), then ( G ) is OWL DL</td>
</tr>
<tr>
<td></td>
<td>- Otherwise, go to step (1)</td>
</tr>
</tbody>
</table>
Limitations of RDFS

Expressivity limitations

- Qualified cardinality restrictions (e.g., no Bicycle $\geq 2$ 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.,
  - $\text{Class}(A \text{ partial restriction(hasB someValuesFrom(C))}$
  - 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 $\mathcal{SHOIN}(D)$
- 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 ($\Rightarrow$ OWL 2 DL) and an RDF-based semantics ($\Rightarrow$ 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

- **Language**: properties of properties
  - property chains \((\text{ObjectPropertyChain}, \text{e.g.}: \text{SubObjectPropertyOf}(\text{ObjectPropertyChain}(\text{a:hasMother, a:hasSister}) \text{a:hasAunt}))\)
  - with having Lois as the mother of Stewie, and Carol a sister of Lois, the ontology entails that Stewie has Carol as aunt
  - \(\text{ObjectMinCardinality}, \text{ObjectMaxCardinality}, \text{ObjectHasSelf}, \text{ObjectExactCardinality}, \text{ObjectHasSelf}, \text{FunctionalObjectProperty}, \text{InverseFunctionalObjectProperty}, \text{DisjointObjectProperties}\) 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

- **Language**: other extensions
  - qualified cardinality restrictions
  - \(\text{The flatkey} \text{’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
  - \(\text{Richer datatypes, data ranges; e.g., DatatypeRestriction(}\text{xsd:integer, xsd:minInclusive “5”}&\text{xsd:integer, xsd:maxExclusive “10”}&\text{xsd:integer)}\)

### Partial table of features

<table>
<thead>
<tr>
<th>Language</th>
<th>OWL 1</th>
<th>OWL 2</th>
<th>OWL 2 profiles</th>
<th>Reasoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>HasKey</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Complement of roles</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Complement of concepts</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Functional dependency</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>One-of, enumerated classes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Qualified number restrictions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Role values</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Symmetry</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Role acyclicity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Role chaining</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>N-ary roles (where (n))</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Role hierarchy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Concept identification</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Complement of roles</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Complement of concepts</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Role acyclicity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Role chaining</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Role hierarchy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Rationale

- **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

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)
### Limitations of RDFS

- OWL Limitations
- OWL 2
- OWL 2 profiles
- Reasoning

### OWL 2 EL Overview

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

- Query answering over a large amount of instances with the 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 $\mathcal{DL}$-$\mathcal{R}_{\mathbb{L}}$ (more is possible with UNA and in some implementations)

---

### 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

- Expressive features include several used features of UML Class diagrams and ER models ('COnceptual MOdel-based Data Access')
- Based on $\mathcal{DL}$-$\mathcal{R}_{\mathbb{L}}$ (more is possible with UNA and in some implementations)
Supported Axioms in OWL 2QL

- There are some 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.
- #∀ and ¬ on lhs, and ∃ and ⊔ on rhs of ⊑

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
- 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 $K$.

### Summary

- **Limitations of RDFS**
  - OWL Design of OWL OWL and Description Logics OWL Syntaxes
- **Limitations**
  - OWL 2
    - OWL 2 DL
    - OWL 2 profiles
      - OWL 2 EL
      - OWL 2 QL
      - OWL 2 RL
- **Reasoning**

## Part III

**Foundational and top-down aspects of ontology engineering**

- **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)*
DOLCE's basic relations
- Parthood
  - Between quality regions (immediate)
  - Between arbitrary objects (temporary)
- Dependence: Specific/generic constant dependence
- Constitution
- 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)
- Participation
- Representation

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

DOLCE's basics on universals
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.

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
  - A Span ontology of happenings and occurrences 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

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 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)

Examples
Overview

- BFO 1.1 in OWL with 39 classes, no object or data properties, in ALC.
- There is a bfo-ro.owl to integration relations of the Relation Ontology with BFO
- 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 Core

- A non-extensional temporal mereology with collections, sums, and universals
- BFO as a collection of smaller theories
  - EMR, QSizeR, RBG, QDiSizeR, ..., Adjacency, Collections, SumPartitions, Universals, Instatiation,
    ExtensionsOfUniversals, PartonomicInclusion, UniversalParthood

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
- E.g., within the OBO Foundry: the “Relation Ontology” with 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
Some questions and problems (not exhaustive...)

- 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?

Analysis of the issues from diverse angles

- Mereological theories (Vareni, 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 modeling and ontology engineering
- Subject domains: thus far, mainly geo, bio, medicine

### Foundational Ontologies

- Parthood relations
- Ontology Design Patterns

<table>
<thead>
<tr>
<th>Foundational Ontologies</th>
<th>Parthood relations</th>
<th>Ontology Design Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Proper parthood&quot;</td>
<td>∀x,y(part_of(x,y))</td>
<td></td>
</tr>
<tr>
<td>&quot;Irreflexivity&quot;</td>
<td>∀x¬(part_of(x,x))</td>
<td></td>
</tr>
<tr>
<td>&quot;Transitivity&quot;</td>
<td>∀x,y,z(part_of(y,z)∧part_of(y,x))→part_of(y,x)</td>
<td></td>
</tr>
<tr>
<td>&quot;Antisymmetry&quot;</td>
<td>∀x,y((part_of(x,y)∧part_of(y,x))→x≡y)</td>
<td></td>
</tr>
</tbody>
</table>

Ground Mereology

- Reflexivity (everything is part of itself)
- Antisymmetry (two distinct things cannot be part of each other, or, if they are, then they are the same thing)
- Transitivity (if x is part of y and y is part of z, then x is part of z)
- Proper parthood

<table>
<thead>
<tr>
<th>Foundational Ontologies</th>
<th>Parthood relations</th>
<th>Ontology Design Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Proper parthood&quot;</td>
<td>∀x,y(proper_part_of(x,y)≡part_of(x,y)∧¬part_of(y,x))</td>
<td></td>
</tr>
<tr>
<td>&quot;Asymmetry&quot;</td>
<td>∀x,y(part_of(x,y)→¬part_of(y,x))</td>
<td></td>
</tr>
<tr>
<td>&quot;Irreflexivity&quot;</td>
<td>∀x¬(part_of(x,x))</td>
<td></td>
</tr>
</tbody>
</table>

Defining other relations with part_of

- Overlap (x and y share a piece z)
- Underlap (x and y are both part of same z)
- Over- & undercross (cross/underlap but not part of)
- Proper overlap & Proper underlap

<table>
<thead>
<tr>
<th>Foundational Ontologies</th>
<th>Parthood relations</th>
<th>Ontology Design Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Overlap&quot;</td>
<td>∀x,y(overlap(x,y)≡∃z(part_of(x,z)∧part_of(y,z)))</td>
<td></td>
</tr>
<tr>
<td>&quot;Underlap&quot;</td>
<td>∀x,y(underlap(x,y)≡∃z(part_of(x,z)∧part_of(y,z)))</td>
<td></td>
</tr>
<tr>
<td>&quot;Over- &amp; undercross&quot;</td>
<td>∀x,y(overcross(x,y)≡overlap(x,y)∧¬part_of(y,x))</td>
<td></td>
</tr>
<tr>
<td>&quot;Proper overlap &amp; Proper underlap&quot;</td>
<td>∀x,y(p_overlap(x,y)≡overcross(x,y)∧overcross(y,x))</td>
<td></td>
</tr>
<tr>
<td>&quot;Proper underlap&quot;</td>
<td>∀x,y(p_underlap(x,y)≡undercross(x,y)∧undercross(y,x))</td>
<td></td>
</tr>
</tbody>
</table>
Ground Mereology

- Minimal Mereology
- Extensional Mereology
- Closure Mereology
- Extensional Closure Mereology

General Extensional Mereology

- Strong supplementation [EM]
  \[ \neg \text{part}_o(y, x) \rightarrow \exists z(\text{part}_o(z, y) \land \neg \text{overlap}(z, x)) \]  (14)
- And add unrestricted fusion [GEM]. Let \( \phi \) be a property or condition, then for every satisfied \( \phi \) there is an entity consisting of all entities that satisfy \( \phi \). Then:
  \[ \exists x (\neg \phi \rightarrow \exists y (\text{overlap}(y, x) \rightarrow \exists z (\phi \land \text{overlap}(y, x)))) \]  (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)

\[ \text{Irreflexivity} \]
\[ \text{Asymmetry} \]
\[ \text{Transitivity} \]

\[ \Rightarrow \]
\[ \Downarrow \]

Fig. 1: Hasse diagram of mereological theories; from weaker to stronger. Going uphill after [44].

Things are improving...

- Early days (90s) and simplest options: DL-role \( R \) as partof, or has-part added as primitive role as \( \supset \), 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):
  \[ \supset \equiv (\text{primitive-part})^* \]  (16)
  \[ \text{Car} \equiv \exists x (\text{Wheel} \sqcap \exists y \text{.Tire}) \]  (17)

Then Car \( \sqsubseteq \exists \text{Tire}\)

- 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 DL\( L_{\mu} \), \( \mathcal{SHOIN} \) and \( \mathcal{SROIQ} \). *: properties of the parthood relation (in M), \( \dagger \): properties of the proper parthood relation (in M).

<table>
<thead>
<tr>
<th>Language</th>
<th>DL( L_{\mu} )</th>
<th>( \mathcal{SHOIN} )</th>
<th>( \mathcal{SROIQ} )</th>
<th>DL-LiteA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>( \dagger )</td>
<td>( \dagger )</td>
<td>( \dagger )</td>
<td>( \dagger )</td>
</tr>
<tr>
<td>Reflexivity</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Antisymmetry</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Transitivity</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Asymmetry</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Irreflexivity</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Acyclicity</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Definitions in OBO Relations Ontology

- Instance-level relations
  - \( \text{part}_o(c_1, t) \in R \): a primitive relation between two continuant instances and a time at which the one is part of the other
  - \( \text{part}_p(p_1, r \text{part}_o n_1) \): 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)
  - \( \text{continued}_c(c_1, t) \in \dagger \): \( \text{located}_c(c_1, t) \) at \( t \) and not \( \text{overlap}(c_1, t) \)
  - \( \text{located}_c(c_1, t) \in \dagger \): a primitive relation between a continuant instance, a spatial region which it occupies, and a time
Definitions in OBO Relations Ontology

- Class-level relations
  - $C \text{ part of } C_1 \iff \text{ for all } c, t, \text{ if } Cct \text{ then there is some } c_1 \text{ such that } C_1c_1t$ and $c \text{ part of } c_1$ at $t$.
  - $P \text{ part of } P_1 \iff \text{ for all } p, \text{ if } Pp \text{ then there is some } p_1 \text{ such that } P_1p_1 \text{ and } p \text{ part of } p_1$.
  - $C \text{ contained in } C_1 \iff \text{ for all } c, t, \text{ if } Cct \text{ then there is some } c_1 \text{ such that } C_1c_1t$ and $c \text{ contained in } c_1$ 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 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 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

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

∀x,y(member_of(x,y) ≜ part_of(x,y) ∧ (POB(x) ∨ SOB(y)) ∧ SOB(y))

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

∀x,y(constitutes(x,y) ≜ constituted_of(y,x) ≜ part_of(x,y) ∧ POB(y) ∧ M(x))

Part-whole relations processes and sub-processes (e.g., Chewing is involved in the grander process of Eating)

∀x,y(involved_in(x,y) ≜ part_of(x,y) ∧ PD(x) ∧ 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)

∀x,y(contains_in(x,y) ≜ part_of(x,y) ∧ R(x) ∧ R(y)) ∧
∃z,w(has_2D(z,x) ∧ has_3D(w,y) ∧ ED(z) ∧ ED(w)))

∀x,y(located_in(x,y) ≜ part_of(x,y) ∧ R(x) ∧ R(y)) ∧
∃z,w(has_2D(z,x) ∧ has_2D(w,y) ∧ ED(z) ∧ ED(w)))

∀x,y(s_part_of(x,y) ≜ part_of(x,y) ∧ ED(x) ∧ ED(y))
Using the taxonomy of part-whole relations

- Representing it correctly in ontologies and conceptual data models
- Reasoning with a taxonomy of relations

Decision diagram

Example - before

- Envelope is not involved-in, not a member-of, does not constitute, is not a sub-quantity of, does not participate-in, is not a geographical object, but instead is contained in the ConferenceBag.
- Transitivity holds for the mereological relations: derived facts are automatically correct, like RegistrationReceipt contained in ConferenceBag.
- Intransitivity of Linen and ConferenceBag, because a conference bag is not wholly constituted of linen (the model does not say what the Flap is made of).
- Completeness, i.e. that all parts make up the whole, is implied thanks to the closed-world assumption. ConferenceBag directly contains the conference documents and Envelope only, and does not contain, say, the Flap.
Foundational ontologies Parthood relations Ontology Design Patterns

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.

The RBox Compatibility service – definitions

Definition (Domain and Range Concepts)

Let $R$ be a role and $R \subseteq C_1 \times C_2$ its associated Domain & Range axiom. Then, with the symbol $D_R$ we indicate the User-defined Domain of $R$—i.e., $D_R = \{c_1 \mid c_1 \in C_1, (c_1, c_2) \in R\}$—while with the symbol $R_R$ we indicate the User-defined Range of $R$—i.e., $R_R = \{c_2 \mid c_2 \in C_2, (c_1, c_2) \in R\}$.

Definition (RBox Compatibility)

For each pair of roles, $R, S$, such that $(T, R) \models R \subseteq S$, check:

1. **Test 1.** $(T, R) \models R \subseteq D_R$ and $(T, R) \models R_R \subseteq D_R$;
2. **Test 2.** $(T, R) \not\models D_S \subseteq D_R$;
3. **Test 3.** $(T, R) \not\models R_R \subseteq R_R$.

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 $R \subseteq D'_R \times R_R$, where $D'_R \equiv D_R \cap D_S$, which subsequently has to go through the RBox compatibility service (and similarly when Test 1 fails on range restrictions).

---

*The axiom $C_1 \equiv C_3$ is a shortcut for the axioms: $C_1 \subseteq C_3$ and $C_3 \subseteq C_1$.*
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

Post-script: extensions in various directions

• Mereotopology, with location, GIS, Region Connection Calculus (http://www.comp.leeds.ac.uk/qsr/rcc.html)

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

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.

Reasoning OPs

- Applications of Logical OPs oriented to obtain certain reasoning results, based on the behavior implemented in a reasoning engine.
- Examples of Reasoning OPs include: classification, subsumption, inheritance, materialization, and de-anonymizing.
- Inform about the state of that ontology, and let a system decide what reasoning has to be performed on the ontology in order to carry out queries, evaluation, etc.
- Name all relevant classes, so no anonymous complex class descriptions are left (restriction deanonymizing). Name anonymous individuals (skolem de-anonymizing). Materialize the subsumption hierarchy (automatic subsumption) and normalize names, Instantiate the deepest possible class or property. Normalize property instances (property value materialization).

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.
- For English language only, thus far.
- Similar to idea of specification of ORM’s verbalization templates.

How to create an ODP

- See chapter 3 of (Presutti et al., 2008).
- From 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

Foundational ontologies
  DOLCE
  BFO

Parthood relations
  Parts, mereology, meronymy
  Taxonomy of types of part-whole relations
  Using the taxonomy of part-whole relations

Ontology Design Patterns
  Types of patterns
  Developing and using an ODP

Part IV

Bottom-up ontology development

Outline

RDBMSs and other ‘legacy KR’
  Example: manual and automated extractions

Natural language
  Introduction
  Ontology learning
  Ontology population

Biological models and thesauri
  Models in biology
  Thesauri

Bottom-up

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

A few languages

Levels of ontological precision

preference: The ability to catch all and only the intended meaning
  (for a logical theory, to be satisfied by intended models)

(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

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

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)

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
Manual mapping to DL-Lite

- 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 (more about that in the next course)

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

Semantic Tagging—Classes, Terms

Examples (out of many)

- GoPubMed (Ging et al, 2008)
  - Layer over PubMed, which indexes > 10mln 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 Fernandez, 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

Semantic Tagging—Lexicalized Ontologies

Examples (out of many)

- GoPubMed (Ging et al, 2008)
  - Layer over PubMed, which indexes > 10mln 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 Fernandez, 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

http://opf.diku.dk/illustration/
http://ontoware.org/projects/jexonto/

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

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, ontology, hyponym structure of WordNet, Haust patterns
  - Termine: statistics of candidate term, such as total frequency of occurrence, frequency of the term as part of 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 (cont’d)

<table>
<thead>
<tr>
<th>LMO terms predicted by TFIDF</th>
<th>LMO terms predicted by RelFreq</th>
</tr>
</thead>
<tbody>
<tr>
<td>890</td>
<td>890</td>
</tr>
<tr>
<td>15.92%</td>
<td>15.92%</td>
</tr>
<tr>
<td>30.32%</td>
<td>30.32%</td>
</tr>
<tr>
<td>28.98%</td>
<td>28.98%</td>
</tr>
<tr>
<td>71.42%</td>
<td>71.42%</td>
</tr>
</tbody>
</table>

From Alexandropoulos et al., 2008

Results

- OntoLearn excluded from 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

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., ‘prevalence 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 (±25% of the TFIDF top50) or can be added to LMO (±46% 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 are 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 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, introduction
- 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 Protege; 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

RDBMSs and other 'legacy KR'

Biological models and thesauri

PathwayAssist vocabulary

Kindly provided by Kristina Hettne

226/368

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

227/368

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

228/368

'Translation' w.r.t. DOLCE categories

- Basic mapping to DOLCE categories:
  - ∀x((Stock(x) ← Entity(x)) → ED(x))
  - ∀x((Flow(x) ← Entity(x)) → PD(x))
  - ∀x((Converter(x) ← Entity(x)) → (Q(x) ∨ ST(x)))
  - ∀x((ActionConnector(x,y) ← Relationship(x,y)))
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

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

The serialized version of the ontology (section)

Section in ezOWL
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

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.)

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

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)

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)
Summary

RDBMSs and other ‘legacy KR’
Example: manual and automated extractions

Natural language
Introduction
Ontology learning
Ontology population

Biological models and thesauri
Models in biology
Thesauri

Part V
Methods and methodologies

Outline

Parameters and dependencies
Example methods: OntoClean and Debugging
Guidance for modelling: OntoClean
Debugging ontologies

Methodologies and tools

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

Purposes

• Querying data by means of an ontology (OBDA) through linking databases to an ontology
• Data(base) 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

Parameters and dependencies
Example methods: OntoClean and Debugging

Methodologies and tools

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?
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., 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.

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;
  (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
  ⇒ 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 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;
- Ontological reasoning services (OntoClean, RBox reasoning service)
- Other technologies (e.g., Bayesian networks)

OntoClean overview

- Problem: messy taxonomies on what subsumes what
- How to put them in the right order?
- OntoClean provides guidelines for this [see 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

- 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)
Parameters and dependencies

Example methods: OntoClean and Debugging Methodologies and tools

Basics

Definition

A non-rigid property carries an IC \( \Gamma \) if it is subsumed by a rigid property carrying \( \Gamma \).

Definition

A property \( \phi \) supplies an IC \( \Gamma \) iff i) it is rigid; ii) it carries \( \Gamma \); and iii) \( \Gamma \) is not carried by all the properties subsuming \( \phi \). This means that, if \( \phi \) 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

- \( +O \) \: Type
- \( +I \) \: Sortal
- \( +R \) \: Non-Sortal
- \( -D \) \: Material role
- \( -D \) \: Mixin
- \( -D \) \: Formal role
- \( -D \) \: Category
- \( -D \) \: Attribution

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 they 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.

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. ‘undesirable’ inferred subsumptions.
- Inconsistent ontologies: all classes taken together unsatisfiable.

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 its 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.

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).

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.
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”: 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

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.

Scenarios for Building Ontology Networks

- 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).

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.
Part VI
Extra topics

Uncertainty and Vagueness

Challenges
Sample applications

Outline

Uncertainty and Vagueness

Uncertain knowledge
Probabilistic logic and ontologies
Possibilistic logic
Vague Knowledge
Many-valued logics and ontologies
Rough sets and ontologies
Tools and applications

Challenges
Modelling the subject domain
Reasoning scenarios
Social Aspects

Sample applications
Semantic Scientific Workflows
More bio-ontologies
Linked data using ontologies
Linking technologies

Examples

- Information Retrieval: To which degree is a Web site, a Web page, a text passage, an image region, a video segment, . . . relevant to my information need?

- Matchmaking: To which degree does an object match my requirements? e.g., your budget is about 20.000 euro to buy a car, then to which degree does a cars price of 20.500 euro match your budget?

- Ontology alignment: To which degree do two concepts of two ontologies represent the same thing, or are disjoint, or are overlapping?

- Classifying ripe apples or “the set of all individuals that mostly buy low calorie food”

Problems: what and how to incorporate such vague or uncertain knowledge in OWL and its reasoners?

Solutions:

i. probabilistic, possibilistic, fuzzy, rough extensions to the language

ii. for reasoning: transform back into OWL and use standard reasoner or develop your own one

Usage, among others:

- Information retrieval (e.g., top-k retrieval)
- Classifying patients (e.g., patients that are possibly septic have properties: infection and (temperature > 38°C OR temperature < 36°C, respiratory rate > 20 breaths/minute OR PaCO2 < 32 mmHg, etc)
- Recommender systems (user preferences etc.)
- Matchmaking in web services

Some of the following slides are taken from Umberto Straccia’s AAAI’07 tutorial [http://gaia.isti.cnr.it/~straccia/download/papers/VANCOUVER07/VANCOUVER07.pdf]

Uncertainty and Vagueness

Uncertainty: statements are true or false, but due to lack of knowledge we can only estimate to which probability / possibility / necessity degree they are true or false

E.g.: a bird flies or does not fly. The probability / possibility / necessity degree that it flies is 0.83

Vagueness: statements involve concepts for which there is no exact definition, such as tall, small, close, far, cheap, expensive, true to some degree, taken from a truth space

E.g., “Hotel Verdi is close to the train station to degree 0.83”

Uncertainty and Vagueness: “It is probable to degree 0.83 that it will be hot tomorrow”

Imperfect information covers notions such as uncertainty, vagueness, contradiction, incompleteness, imprecision
Probabilistic logic: Syntax

- Finite nonempty set of basic events $\Phi = \{p_1, \ldots, p_n\}$, with $n \geq 1$
- Events: every element of $\Phi \cup \{\top, \bot\}$ is an event; if $\phi$ and $\psi$ are events, then so are $\neg \phi$, $(\phi \land \psi)$, $(\phi \lor \psi)$, and $(\phi \rightarrow \psi)$
- A probabilistic formula is an expression of the form $\phi \geq l$, with $l \in \mathbb{R}$ from the unit interval $[0, 1]$ (note that $\neg \phi \geq 1 - u$ encodes $\phi$ is true with probability at most $u$)
- Conditional constraint $\langle \psi \mid \phi \rangle[I, u]$: events $\psi$ and $\phi$, and $l, u \in [0, 1]$, which denotes “the conditional probability of $\psi$ given $\phi$ is in $[l, u]$"
- Probabilistic knowledge base $\mathcal{KB} = (\mathcal{L}, \mathcal{P})$
  - finite set of logical constraints $\mathcal{L}$
  - finite set of conditional constraints $\mathcal{P}$

Probabilistic logic: Semantics

- A world $I$ associates with every basic event in $\Phi$ a binary truth value, and extend $I$ by induction to all events as usual
- $I_{\Phi}$ is the (finite) set of all worlds for $\Phi$
- $I_{\Phi}$ is an event if $I$ is a model of $\phi$, denoted $I \models \phi$ iff $I(\phi) = true$
- Probabilistic interpretation $Pr$: probability function on $I_{\Phi}$ s.t. all $Pr(I)$ with $I \in I_{\Phi}$ sum up to 1
- $Pr(\phi)$ is the sum of all $Pr(I)$ such that $I \in I_{\Phi}$ and $I \models \phi$
- $Pr(\psi \mid \phi)$: if $Pr(\phi) > 0$, then $Pr(\psi \mid \phi) = \frac{Pr(\psi \land \phi)}{Pr(\phi)}$

Probabilistic RDF, OWL, and DLs

- P-SHOQ(D), P-SHOIN(D) (by T. Lukasiewicz)
  - uses the notion of a conditional constraint
  - semantics is based on the notion of lexicographic entailment in probabilistic default reasoning
- probabilistic TBox and ABox probabilistic knowledge as statistical knowledge and as degrees of belief about instances of concepts and roles, respectively
  - allows for deriving both statistical knowledge and degrees of belief
  - allows for expressing default knowledge about concepts
- PR-OWL (by da Costa and Laskey)
  - Probabilistic semantics based on multi-entity Bayesian networks
- And others with Bayesian networks, with DLs, covering various permutations of probabilistic KR&R added to different languages

Use of Probabilistic Ontologies

- Representation of terminological and assertional probabilistic knowledge (e.g., in the medical domain or at the stock exchange market)
- Information retrieval, for an increased recall
- Ontology matching
- Probabilistic data integration, especially for handling ambiguous and controversial pieces of information

Possibilistic logic introduction

- Syntactically, use possibilistic formulas to constrain the necessities and possibilities of propositional events
- Semantically, possibility distributions on worlds, each of which associates with every event a unique possibility and a unique necessity
- Possibility of an event is the maximum of the possibilities of all worlds that satisfy the event
- Useful for encoding user preferences, since possibility measures can be viewed as rankings (on worlds or also objects) along an ordinal scale
- While reasoning in possibilistic logic generally requires to solve linear optimization problems, reasoning in possibilistic logic does not and thus can generally be done with less computational effort
Possibilistic logic: Syntax and Semantics

- Possibilistic formulae have the form $P \phi \geq l$ or $\neg P \phi \geq l$, with $\phi$ event, $l \in \mathbb{R}$ from $[0, 1]$. Possibly, and Necessarily, e.g.: $P \neg \text{snow\_today} \geq 0.7$ encodes that it will snow today is possible to degree 0.7
- $N \text{mother} \rightarrow \text{female} \geq 1$ says that a mother is necessarily female
- A possibilistic formula is a pair $(\phi, n)$ consisting of a classical logic formula $\phi$ and a degree $n$ expressing certainty or priority (which also can be considered as possibility degree of $\phi$)
- A possibilistic knowledge base $KB$ is a finite set of possibilistic formulae, of the form $KB = \{ (\phi, n): i = 1 \ldots n \}$
- A possibilistic interpretation is a mapping $\pi: I_\phi \rightarrow [0, 1]$
  - $\pi(I)$ is the degree to which world $I$ is possible
  - every world $I$ such that $\pi(I) = 0$ is impossible
  - every world $I$ such that $\pi(I) = 1$ is totally possible
  - $\pi$ is normalized iff $\pi(I) = 1$ for some $I \in I_\phi$

Possibilistic ontologies

- Add it to an arbitrary DL language (including any of the DL-based OWL languages)
- E.g. Qi, Pan, Ji in DL’07, supposedly with basics implemented in KAON2
- Possibilistic generalization of $\text{ALC}$ for information retrieval (Liu and Yao, 2001), used for query relaxation, restriction, and exemplar-based retrieval
- Thus far: little usage, examples are toy examples

Introduction

- Vagueness: statements involve concepts for which there is no exact definition, such as tall, close
- Statements are true to some degree which is taken from a truth space, which are usually $[0, 1]$
  - Hotel Verdi is close to the train station to degree 0.83
  - Find top-k cheapest hotels close to the train station:
    - $q(h) \leftarrow \text{hasLocation }(h, \text{hl}) \land \text{hasLocation }(\text{train}, \text{cl}) \land \linebreak \text{close}(\text{hl, cl}) \land \text{cheap}(h)$
    - What is the interpretation of $\text{close(Verdi, train)} \land \text{cheap(200)}$?
    - Interpretation: a function $I$ mapping atoms into $[0, 1]$, i.e. $I(A) \in [0, 1]$
    - If $I(\text{close(Verdi, train)}) = 0.83$ and $I(\text{cheap(200)}) = 0.2$, then what is the result of $0.83 \land 0.2$?
- More generally, what is the result of $n \land m$, for $n, m \in [0, 1]$?

Fuzzy logic (basics)

- Formulae: First-Order Logic formulae, terms are either variables or constants
- many-valued formulae have the form $\alpha \geq l$ or $\neg \alpha \leq u$ where $l, u \in [0, 1]$ (degree of truth is at least $l$ and at most $u$, resp.)
- Formulae have a degree of truth in truth space $[0, 1]$
- Interpretation is a mapping $I: \text{Atoms} \rightarrow [0, 1]$, which are extended to formulae as follows (subsections):
  - $I(\neg \phi) = I(\phi) \rightarrow 0$ (18)
  - $I(\exists x \phi) = \sup_{e \in E} I^*_e(\phi)$ (19)
  - $I(\forall x \phi) = \inf_{e \in E} I^*_e(\phi)$ (20)
  - $I(\phi \land \psi) = I(\phi) \land I(\psi)$ (21)
  - $I(\phi \lor \psi) = I(\phi) \lor I(\psi)$ (22)
  - $I(\phi \rightarrow \psi) = I(\phi) \rightarrow I(\psi)$ (23)
  - $I(\neg \phi) = I(\phi)$ (24)
Uncertainty and vagueness

- Challenges
- Sample applications

305/368

- Uncertainty and vagueness
- Challenges
- Sample applications

cont’d

- where $I_x$ is as $I$ except that var $x$ is mapped to individual $c$
- $\otimes$, $\oplus$, $\rightarrow$, and $\neg$ are combination functions: triangular norms (ar-t-norms), triangular co-norms (or s-norms), implication functions, and negation functions, respectively
- which extend the classical Boolean conjunction, disjunction, implication, and negation, respectively, to the many-valued case
- Degree of subsumption between two fuzzy sets $A$ and $B$, denoted $A \subseteq B$, is defined as $\inf_x I_x(A(x)) = B(x)$
  - if $A(x) \leq B(x)$ for all $x \in [0,1]$ then $A \subseteq B$ evaluates to 1
- $I$ $\triangleright$ $\geq 1$ (resp. $I$ $\triangleright$ $\leq u$) $\triangleright$ $\geq 1$ (resp. $I$ $\triangleright$ $\leq u$).

302/368

Fuzzy DLS can be classified according to:

- The description logic resp. ontology language that they generalize
- The allowed fuzzy constructs and the underlying fuzzy logics (Gödel, Lukasiewicz, Zadeh, ...)
- Their reasoning services:
  - Consistency, Subsumption, Equivalence
  - Graded instantiation: Check if individual $x$ is an instance of class $C$ to degree at least $n$, i.e. $KB \triangleright (x : C \geq n)$
  - Best Truth Value Bound problem: determine the greatest lower bound $n \in [0,1]$ of an axiom $A$, i.e. $glb(KB, A) = \sup(n \mid KB \triangleright (x \geq n))$ (likewise for lub)
  - Best Satisfiability Bound problem: $glb(KB, C)$ determined by the max value of $x \in \mathbb{R}$ s.t. $[R; T, A(x \geq C \geq x)]$ (among of models), determine the max degree of truth that concept $C$ may have over all individuals $x \in \mathbb{R}$

- $glb(KB, C \subseteq D)$ is the minimal value of $x$ such that $KB \triangleright (R; T, A(x : C \geq D \geq x))$ is satisfiable, where $x$ is a new individual. Therefore, the greatest lower bound problem can be reduced to the minimal satisfiability problem of a fuzzy knowledge base

303/368

Concrete fuzzy concepts

- Examples: Small, Young, Tall etc, with explicit membership function
- Use concrete domains to specify them:
  - $D = (\Delta_D, \Phi_D)$, where $\Delta_D$ is an interpretation domain and $\Phi_D$ the set of concrete fuzzy domain predicates $D$ with a predefined $\Phi_D$ $= \{D \mid n = 1, 2$ and fixed interpretation $\Phi_D$ $\}$
- For instance: $\geq 18(x)$ over $N$, evaluates to true if $x \leq 18$, false otherwise, or $n(0,18)$
- Define $\text{Min} = \text{Person} \sqcap \exists \text{Age} ; \geq 18$
- Let $\text{Young} : \text{Natural} \rightarrow [0,1]$ be a fuzzy datatype predicate denoting the degree of youngness
- Define $\text{Young}(x) = \max(x, 0.30)$, where is the usual left shoulder function
- Define $\text{YoungPerson} = \text{Person} \sqcap \exists \text{Age} ; \text{Young}$
- Then, the $KB$ entails, e.g.:
  - $KB \triangleright \text{Min} \subseteq \text{YoungPerson} \geq 0.6$
  - $\text{YoungPerson} \subseteq \text{Min} \geq 0.4$

304/368

Modifiers

- Very, moreOrLess, slightly, etc.
- Apply to fuzzy sets to change their membership function
  - fuzzy modifier $m$ represents a function $f_m : [0,1] \rightarrow [0,1]$, with $M$ an alphabet for fuzzy modifiers and $m \in M$
  - then, if $C$ is a concept in, say, fuzzy $\mathcal{SHIF}(D)$, then so is $m(C)$
- Modifiers are definable as linear in-equations over $\mathbb{Q}$, $\mathbb{Z}$ (e.g., linear hedges), for instance, linear hedges, for instance, linear hedges $11$, $\inf(x; a, b)$. e.g. very $= \inf(x; 0.7, 0.49)$

- Example:
  - $f_{\text{smoother}}(x) = x^2$
  - $f_{\text{slowdown}}(x) = \sqrt{x}$
  - $\text{SportsCar} \sqsubseteq \text{Car} \sqcap \exists \text{speed} \neg \text{very}(\text{High})$, where very is the fuzzy modifier and High a fuzzy datatype over the domain of speed (in km/h) and may be defined as, say, $\text{High}(x) = \frac{x}{(80, 250)}$

$11$ May modify the shape of a fuzzy set in predictable ways, e.g., by pushing all values less than one towards zero, thereby shrinking the fuzzy part of the set closer to the area that is completely in the set

305/368

Fuzzy OWL

- $\mathcal{SHIF}(D)$, $\mathcal{SHOIN}(D), \mathcal{SROIQ}(D)$, ...
- Additionally, we add
  - modifiers (e.g., very)
  - concrete fuzzy concepts (e.g., Young)
- both additions have explicit membership functions

306/368

The car seller and buyer revisited

- More fine grained approach: prices as vague constraints
  - Seller would sell above 31500 euro, but can go down to 30500
  - Buyer prefers to spend less than 30000, but can go up to 32000
- $\text{AudiTT} \sqcap \exists \text{hasPrice} : (30000, 32000)$
- $\exists \text{hasPrice} : (30000, 32000)$
- with $\text{hasPrice}$ right-shoulder function and is the left-shoulder function
- Highest degree to which $C = \text{AudiTT} \sqcap \exists \text{Request}$ is satisfiable is 0.75 (possibility that the Audi TT and the query matches is 0.75; the $glb(KB, C) = 0.75$)
- the car may be sold at 31250 euro
Uncertainty and vagueness

Challenges

Sample applications

Rough sets (cont’d)

• From the objects in universe U, we want to represent set X such that \( X \subseteq U \) using the attribute set \( P \subseteq A \)

• \([x]_P\) denotes the equivalence classes of the \( p \)-indistinguishability relation

• \( X \) may not be represented in a crisp way—the set may include and/or exclude objects which are indistinguishable on the basis of the attributes in \( P \)—but it can be approximated by using lower and upper approximation, respectively:

\[
PX = \{ x | [x]_P \subseteq X \} \tag{26}
\]

\[
PX = \{ x | [x]_P \cap X \neq \emptyset \} \tag{27}
\]

Rough sets (cont’d)

• The lower approximation is the set of objects that are positively classified as being members of set \( X \), i.e., union of all equivalence classes in \([x]_P\). The upper approximation is the set of objects that are possibly in \( X \)

• Its complement, \( U - \overline{PX} \), is the negative region with sets of objects that are definitely not in \( X \) (i.e., \( \neg X \))

• with every rough set we associate two crisp sets, called lower and upper approximation, denoted as a tuple \( X = (\overline{X}, X) \)

• The difference between the lower and upper approximation, \( BPX = \overline{PX} - PX \), is the boundary region of which its objects neither can be classified as to be member of \( X \) nor that they are not in \( X \); if \( BPX = \emptyset \) then \( X \) is, in fact, a crisp set with respect to \( P \) and when \( BPX \neq \emptyset \) then \( X \) is rough w.r.t. \( P \)

Rough ontologies: Introduction

• Extension of rough sets to the knowledge representation layer; or: extension of crisp concepts in an ontology to incomplete data in the ABox

• Preliminary results with ‘rough ontologies’, feasibility study

• Some results with fuzzy-rough and rough-fuzzy ontology languages

• No (end-user) tools and demonstration cases in a subject domain yet; requires linking of ontology to sufficient data, i.e., need for scalable semantic web technologies

Fuzzy OWL and reasoning

• Three principal approaches tested: Tableaux method, MILP based method, MIQP based method

• Implementation issues; Several options exists:

  • Try to map fuzzy DLs to classical DLs, but difficult to work with modifiers and concrete fuzzy concepts

  • Build ad-hoc theorem prover for fuzzy DLs, using e.g., MILP

  • A theorem prover for fuzzy \( SHIF \) + linear hedges + concrete fuzzy concepts + linear equational constraints + datatypes, under classical, Zadeh, Lukasiewicz and Product t-norm semantics has been implemented (http://gais.inti.it/~attracta)

  • FIRE: a fuzzy DL theorem prover for fuzzy \( SHIF \) under Zadeh semantics (http://www.image.ece.ntua.gr/~nsimou/)

• Brief introduction of the Pawlak rough set model

\[
\begin{align*}
\text{Granule with object(s)} & \quad \text{Set} \quad \text{X} \\
\text{Universe} \ U & \quad \text{Upper approximation} \\
\text{Lower approximation} &
\end{align*}
\]

• I = (U, A) is called an information system, where U is a non-empty finite set of objects and A a finite non-empty set of attributes

• For every \( a \in A \), function \( a : U \mapsto V_a \) where \( V_a \) is the set of values that attribute \( a \) can have

• For any subset of attributes \( P \subseteq A \), one can define the equivalence relation \( \text{IND}(P) \) as

\[
\text{IND}(P) = \{ (x,y) \in U \times U \ | \ \forall a \in P, a(x) = a(y) \} \tag{25}
\]

• \( \text{IND}(P) \) generates a partition of \( U \), which is denoted with \( U/\text{IND}(P) \), or \( U/P \) for short.

• If \( (x,y) \in \text{IND}(P) \), then \( x \) and \( y \) are indistinguishable with respect to the attributes in \( P \), i.e., they are \( p \)-indistinguishable.
Rough ontologies

- Several proposals, mainly DL+ rough extensions
- Commit to different aspects of the semantics of rough sets/roughness (thus also as to what rough concepts and rough ontologies actually are)
- $E$ is the symmetric, reflexive, transitive indistinguishability relation
- Let $C_R$ be a (rough) concept in a DL language, then semantics for its lower and upper approximation are:

\[ C = \{ x \mid \forall y : (x, y) \in E \rightarrow y \in C \} \]
\[ \overline{C} = \{ x \mid \exists y : (x, y) \in E \land y \in \overline{C} \} \]  
(28)

- Interpretation should map every approximate concept $C_R = (C, \overline{C})$ to a pair over $\Delta^2$, i.e., extending $\rightarrow$ as follows:

\[ C_R^I = (\langle C \rangle^I, (\overline{C})^I) \]  
(30)

- Interesting property: $C \sqsubseteq D \Rightarrow (C, \overline{C}) \sqsubseteq (D, \overline{D})$

Reasoning services

- Alike the standard DL reasoning services:
  - approximate concept satisfiability, being the definitely satisfiability and possibly satisfiability (note that if $C_R$ is possibly unsatisfiable, it is also definitely unsatisfiable)
  - approximate concepts rough subsumption reasoning
  - may be reduced to concept satisfiability problem in classical description logics (after transformation from RoughDL to standard DL)
- Instance classification of the objects into the approximations and their corresponding rough concepts

A scenario

Suppose a person would like to “buy a sports car that costs at most about 22 000 euro and that has a power of around 150 HP”

- the buyer has to manually search for car selling web sites, e.g., using Google;
- select the most promising sites;
- browse through them, query them to see the cars that each site sells, and match the cars with the requirements;
- select the offers in each web site that match the requirements;
- eventually merge all the best offers from each site and select the best ones.

Tools

- Probabilistic ontology tools:
  - Pronto: pellet + probabilistic http://pellet.owldl.com/pronto/
  - PR-OWL http://www.pr-owl.org/
  - Probabilistic Ontology Alignment Tool http://gaia.isti.cnr.it/
  - TOSS http://om.unicar.umd.edu/ptos.html

- Fuzzy ontology tools:
  - Fuzzy RDF
  - FuzzyDL
  - FIRE http://www.openclinical.org/prj/banizar.html
Examples

- Probabilistic ontologies:
  - Star Trek ontology (experimental ontology to demonstrate PR-OWL) http://www.pr-owl.org/prowl/ontostartrek.php
  - Astronomy to demonstrate TOSS http://om.umiacs.umd.edu/pparq.html
- Fuzzy ontologies:
  - Oncology with FIRE http://www.oncolor.org/
  - FIRE with an medical imaging example

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 (some of the limitations and extensions discussed in lecture 2, 6 and 7)
- 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

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

- n-ary relations, where $n \geq 2$
  - Hepatitis hasSymptom Fever in most but not all cases
    - What about doing it with probabilistic default knowledge (lecture 7)?
    - $(\forall \phi) \exists l [\phi \land l]$ as “generally, if an object belongs to $\phi$, then it belongs to $l$ with a probability in $[0,1]$”
    - e.g., [HasSymptoms.Fever | 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 bearer of) 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)

- "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 $\square$
    - e.g., $\text{AspirinIntake} \sqsubseteq \square^+\text{preventedBy.MyocardialInfarction}, \text{Or}$
    - $\text{MyocardialInfarction} \sqsubseteq \square^+\text{preventedBy.AspirinIntake}, \text{Or}$
    - $\text{AspirinIntake} \sqsubseteq \square^+\text{hasPhysiologicalEffect.} \cdot \text{MyocardialInfarction}$

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 macrophase 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 xz?”
  ii. “for each x:X, y:Y, r:R, XRY, does there exist an xz2 and an xza where z:Z, s:S, a:A, t:T hold?”
  iii. Find-me-anything-you-have: “for each x:X, return any n1,...,n0, their type of role and the concepts Y1,...,Ye, they are related to”

\[ \begin{array}{ccc}
(i) & (ii) & (iii) \\
X & S & Y \\
A & T & Z \\
R & R & R \\
\end{array} \]

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

Summary

Uncertainty and vagueness

- Uncertain knowledge
- Probabilistic logic and ontologies
- Possibilistic logic
- Vague Knowledge
- Many-valued logics and ontologies
- Rough sets and ontologies

Challenges

- Modelling the subject domain
- Reasoning scenarios
- Social Aspects

Sample applications

- Semantic Scientific Workflows
- More bio-ontologies
- Linked data using ontologies
- Linking technologies

Requirements\(^{12}\)

- Seamless access to resources and service
- Service composition & reuse and workflow design
- Scalability
- Detached execution
- Reliability and fault-tolerance
- User-interaction
- “Smart” re-runs
- “Smart” (semantic) links
- Data provenance

Some general characteristics of Scientific Workflows\(^{13}\)

- The ability to handle many and varied analysis tools; not merely database systems that have to be linked up, but the many (custom-made) analysis tools w.r.t. amount of databases
- Interfaces to a diverse range of computational environments (supercomputers, grid, Internet and Semantic Web)
- The ability to handle activity mixes that are different from typical business profiles—and there are, at least initially, few canned and reusable workflows (i.e., design from scratch)
- Need for explicit representation of knowledge at different stages
- Auditability of the computations (when the results are used to make decisions that carry regulatory or legislative implications; e.g., data analysis of clinical trials, climate model predictions)

\(^{12}\)Ludäscher et al. Scientific Workflow Management and the Kepler System

\(^{13}\)http://people.engr.ncsu.edu/mensing/papers/databases/workflows/sciworkflows.html
Further additions w.r.t. bio and the Semantic Web

- Software version of the “Materials & Methods”, i.e., a ‘pipeline’ of activities that can, is, and has to be, be carried out more than once
- Repeatability of the in silico experiment
- Customization of the data sources and methods for each researcher
- ‘open’ system
- Provenance of the data; or: a need for addressing the trust layer in the Semantic Web layer cake

Where can we plug SWT languages into Scientific Workflows?

- RDF: common data format (for linking and integration)
- SPARQL: querying data
- OWL: representation of the knowledge across the workflow
- Rules: orchestrate the service execution
- Services (e.g., WSDL, OWL-S): to discover useful scripts that can perform a task in the workflow
- The “trust” (provenance) layer: ...... (currently with any of the previous ones)

Provenance and trust, system examples

- Taverna: experiment-, workflow-, and knowledge-provenance, representing a mixture of RDF(S) and OWL to represent the overall model, individual provenance graphs of a particular workflow.\(^\text{14}\)
- PASS experiments, with another provenance ontology for the workflow,\(^\text{15}\), and Pychinko, a Semantic Web rule engine to orchestrate the service execution.\(^\text{16}\)

Shopping for approaches to achieve data integration

I. Physical schema mappings
   - Global As View (GAV)
   - Local As View (LAV)
   - GLAV

II. Conceptual model-based data integration

III. Data federation

IV. Data warehouses

V. Data marts

VI. Services-mediated integration

VII. Peer-to-peer data integration

VIII. Ontology-based data integration
   - I or II (possibly in conjunction with the others) through an ontology
   - Linked data by means of an ontology

\(^\text{15}\)http://provenance.academy.org/provenance.owl
Classification of data integration approaches and tools

Overview

• Ontology on top of physical schemas?
• Ontology on top of conceptual data models
• Ontology to mediate between services
• Classifying instances into an ontology

Linked data in Bio

• Data-level integration
• Annotated instances stored in databases
• Across databases at physically different locations
• On the Web
  Where the ontology tells you which ones are the same, or
  instantiating the same universal represented in the ontology

Integration and annotations examples

• GenMapper
  • Centralised, with a global view
  • Exploits existing mappings between objects/sources
  • Links between the databases through the annotations of the
    objects (e.g., genes, proteins)
  • Links to terms of the ontology (GO), i.e., (semi-)manual
    classification
• Distributed Annotation Systems (DAS)
  • Distributed, mapping-based, no global view
  • Central genome server as primary source that contains the
    reference genome sequence
  • Separately, several annotation servers where the sources are
    wrapped
  • Recalculation of all annotations when the reference sequence
    has changed

Web-links based 'integration'

• Web-Link = URL of a source + ID of the object of interest
• Little integration effort, Scaleable, Navigational analysis: only
  one object at a time
• A mere link is semantics-poor w.r.t. language and subject
  domain meaning, e.g.:
  • How would one do automated reasoning with it to derive
    implicit knowledge? (not)
  • “related to” versus, among others, partOf, isA, containedIn,
    etc; i.e., even poorer than the thesaurus’ RT, BT, NT
  • DBGET + LinkDB
  • see also http://www.genome.jp/dbget/
### Challenges

- Uncertainty and vagueness
- Sample applications

### Sample applications

#### Expressive ontology vs scalability & performance

- Some ontologies in OWL (2007), denoted in their DL language used

| DL Language | OWL 2007
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DOLCE</td>
<td>AE2, NDF(0)</td>
</tr>
<tr>
<td>SHOIN</td>
<td>AE2, NDF(0)</td>
</tr>
<tr>
<td>AL</td>
<td>AE2, NDF(0)</td>
</tr>
<tr>
<td>ALCHIF</td>
<td>AE2, NDF(0)</td>
</tr>
<tr>
<td>ALCHON</td>
<td>AE2, NDF(0)</td>
</tr>
<tr>
<td>SIN</td>
<td>AE2, NDF(0)</td>
</tr>
<tr>
<td>AL</td>
<td>AE2, NDF(0)</td>
</tr>
<tr>
<td>ALF</td>
<td>AE2, NDF(0)</td>
</tr>
<tr>
<td>SHF</td>
<td>AE2, NDF(0)</td>
</tr>
</tbody>
</table>

- "Breakpoint" is known roughly and through disparate experiments, but not (yet) through benchmarking
- Lite-izing ontologies

### Generalising the current bio-integration implementations

- Many CS theory and technologies ‘on offer’ that purport to solve each integration problem
- All of them experimented with by the users, who added linked data, annotations, and web-links to the array of options
- For all: still a lot of manual work
- For all: for various reasons fairly simple end-user level queries (which might well be complicated at the back-end)
- Does it actually solve the original problem and address the requirements as defined by the GOC? (see part 1)
- Ontology usage: ‘simple’ ontologies, or none at all
- Semantic Web Technologies usage: ...

### Other integration systems (examples)

- BioFuice, based on iFuice:
  - Use instance-level cross-references for instance-level mappings between sources
  - Mappings have a semantic mapping type
  - Domain model (an ontology) indicates available object types and relationships
- Sequence Retrieval System: wrapping sources, making them accessible through one interface
- BioGuide: selecting appropriate sources and tools using chosen preferences and strategy
- IMGT-Choreography based on the IMGT-ONTOLOGY concepts to coordinate services among databases
- Mash-ups, RDF, XML, ...

### Background

- Main players in SWLS are engineers, domain experts, bioinformaticians, bio-ontologists. “Something bio” covers many disciplines: e.g., genomics, metabolomics, ecoinformatics, and, above all: biomed & healthcare. Diverse fields, diverse needs.
- Some current characteristics:
  - Collaboration & interdisciplinary work
  - Possible not-intended use of technologies (from the perspective of computer scientist)
  - Novel-ness of the technologies: data integration techniques of the ’90s did not solve the issues, SW tech will?
  - Goal-driven: looking for the “killer app” and discover novel information about nature.
  - Thus far, there are very few success stories

### Queries in the SW

- What can you do? We have:
  - Within the SW-scope, we have: SPARQL, SeRQL, XQuery, XPath, Xcerpt, Prova, ...
  - Know their strengths and weaknesses17, tool support
  - Performance issues (e.g. interval join with several query languages Cell Cycle Ontology browsing)
- But is that what the user wants?
  - Recursive queries
  - Subgraph isomorphisms
  - Query data through the ontology
  - Traverse paths of arbitrary (finite, but not pre-defined) length
- ...

---


---

Semantic Web Technologies for HC & LS

- The Semantic Web will solve all your problems?
- W3C Health and Life Sciences Interest Group
  - “is designed to improve collaboration, research and development, and innovation adoption in the health care and life science industries. Aiding decision-making in clinical research, Semantic Web technologies will bridge many forms of biological and medical information across institutions.”
  - “Subgroups focus on making biomedical data available in RDF, working with biomedical ontologies, prototyping clinical decision support systems, working on drug safety and efficacy communication, and supporting disease researchers navigating and annotating the large amount of potentially relevant literature.”
- Example activity resulting in the BMC Bioinformatics articles “Advancing translational research with the Semantic Web” (2007) and “A journey to Semantic Web query federation in the life sciences” (2009)

From bench to bedside — and from CS theory to software application

Overview 23-author article by Ruttenberg et al., 2007

- “A significant barrier to translational research is the lack of uniformly structured data across related biomedical domains.”
- “Current tools and standards are already adequate to implement components of the bench-to-bedside vision.”
- “Gaps in standards and implementations still exist and adoption is limited by typical problems with early technology... growing pains as the technology is scaled up.”
- “SW ‘will improve the productivity of research, help raise the quality of health care, and enable scientists to formulate new hypotheses inspiring research based on clinical experiences”

What do they want?

- Data integration
- Querying the data across databases
- Expressive ontology languages to represent biological knowledge
- Manage (query) the data silos ('write-only database')
- Building upon the web of data
- Automation to ‘upgrade’ ‘legacy’ material to SemWeb technologies and standards
- Navigate and annotate potentially relevant literature

How do they do it?

- Global scope of identifiers
- RDF/OWL
- Bottom-up development
  - RDF triple stores from ‘legacy’ RDBMS
  - Previously discussed bottom-up techniques for ontology development
  - SWRL for rules

A few discussion questions

- What do they want?
- How do they do it?

A few discussion questions

- Are (should?) “Tools and strategies to extract or translate from non-RDF data sources to enable their interoperability with data organized as statements.” (be) part of the set of SW Technologies?
  - Or: where are (W3C) standardization efforts for RDFBMS—RDF, excel→RDF, OBO—OWL, structured flat file → language y mappings?
  - “BioRDF has the goal of converting a number of publicly available life sciences data sources into RDF and OWL.”
  - Thus: not using SW Tech but preparing for use

- “While the need to integrate more types of data will continue, RDFS and OWL offer some relief to the burden of understanding data schemas.”
  - Since when are ontologies read in their OWL syntax-format (or XML-serialised) human understandable? Did you learn RDFS on a rainy Sunday afternoon?
  - UML, ER, ORM, and conceptual graphs are well-established graphical and formal conceptual data modelling languages, is something wrong with using those ones?
A few discussion questions

- "A goal of the HCLSIG is to facilitate creation, evaluation and maintenance of core vocabularies and ontologies to support cross-community data integration and collaborative efforts. Although there has been substantial effort in recent years to tackle these problems, the methodology, tools, and strategies are not widely known to biomedical researchers."
  - Which "methodology, tools, and strategies"?
  - How would you address the lack of necessary skills of the (presumably intended) user-base of biomedical researchers?
  - "The role of the ontologies task force is to work on well-defined use cases, supporting the other HCLSIG working groups."

Current identified technical limitations

- As listed in the article:
  - Scarcity of semantically annotated information sources
  - Performance and scalability
  - Representation of evidence and data provenance
  - Lack of a standard rule language
  - Did you spot other limitations?

Summary

Uncertainty and vagueness

Uncertain knowledge
- Probabilistic logic and ontologies
- Possibilistic logic

Vague Knowledge
- Many-valued logics and ontologies
- Rough sets and ontologies

Tools and applications

Challenges
- Modelling the subject domain
- Reasoning scenarios
- Social Aspects

Sample applications
- Semantic Scientific Workflows
- More bio-ontologies
- Linked data using ontologies
- Linking technologies