Ontology Engineering

Lecture 8: Bottom-up Ontology Development

Maria Keet

email: mkeet@cs.uct.ac.za

home: http://www.meteck.org

Department of Computer Science
University of Cape Town, South Africa

Semester 2, Block I, 2019

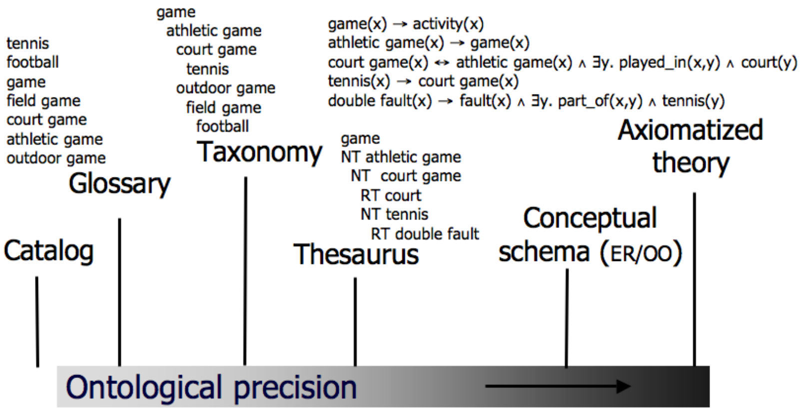
Outline

1. RDBMSs
   - From conceptual model to ontology
   - From data to ontology
2. Thesauri
3. Natural language
   - Introduction
   - Ontology learning and population

Bottom-up

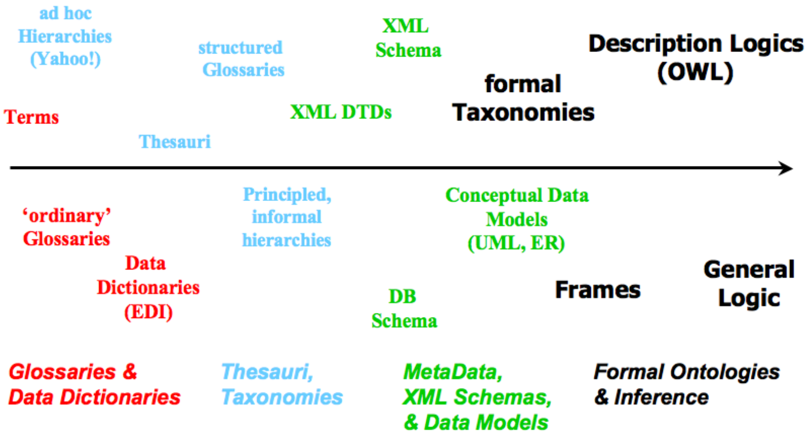
- From *some* seemingly suitable legacy representation to an OWL ontology
  - Database reverse engineering
  - Conceptual model (ER, UML)
  - Frame-based system
  - OBO format
  - Thesauri
  - Formalising biological models
  - Excel sheets
  - Text mining, machine learning, clustering
  - etc...

Levels of ontological precision



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

A few languages

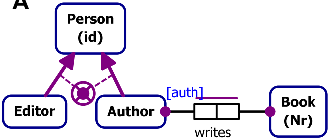


Outline

1. RDBMSs
   - From conceptual model to ontology
   - From data to ontology
2. Thesauri
3. Natural language
   - Introduction
   - Ontology learning and population

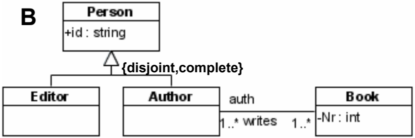
Example models

A

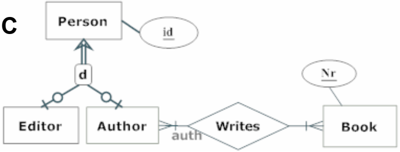


For each Person, **exactly one of the following holds:**
some Author is **that** Person; **some** Editor is **that** Person.
It is possible that more than one Author writes **the same** Book **and that the same** Author writes **more than one** Book.
Each Book, Author **combination occurs at most once in the population of** Author writes Book.
Each Author writes **some** Book.
For each Book, **some** Author writes **that** Book.

B



C



(Re-)using conceptual models

- Recall differences between conceptual models and ontologies (lecture 1)
- We may be able to reuse some of the classes and their associations

(Re-)using conceptual models

- Recall differences between conceptual models and ontologies (lecture 1)
- We may be able to reuse some of the classes and their associations
- First step to address: most of those diagrams are informal, ontologies are logic-based
- (sub step: there are multiple formalisations for UML, ER, ORM, ...; which one to choose, or make a new one?)

Toy example

- Exercise: formalise the example(s) from the previous slide
- Note: you may be lenient to yourself, for now ...

Toy example

- Exercise: formalise the example(s) from the previous slide
- Note: you may be lenient to yourself, for now ...
- The models are actually not exactly the same, notably: attributes, identifiers, DL role components

Toy example

- Exercise: formalise the example(s) from the previous slide
- Note: you may be lenient to yourself, for now ...
- The models are actually not exactly the same, notably: attributes, identifiers, DL role components
- $\texttt{Editor} \sqsubseteq \texttt{Person}$, $\exists \texttt{writes}.\texttt{Book} \sqsubseteq \texttt{Author}$, ..., $\texttt{Author} \sqsubseteq = 1\, \texttt{writes}.\texttt{Book}$ (or $\exists$ with ≤ 1—what difference does it make?), ...

Brushing up

- Generalise from, or remove, the application-specific components
  - e.g.: those part-whole relations w.r.t UML's aggregation association
- Perhaps use a foundational ontology to characterise the candidate classes and object properties
- Could use OntoClean aspects (e.g., with OntoUML)
- Add definitions (defined classes), disjointness where appropriate
- More?

General considerations for RDBMSs

- Assume resolved issues of data duplication, violations of integrity constraints, hacks, outdated imports from other databases, outdated conceptual data models

General considerations for RDBMSs

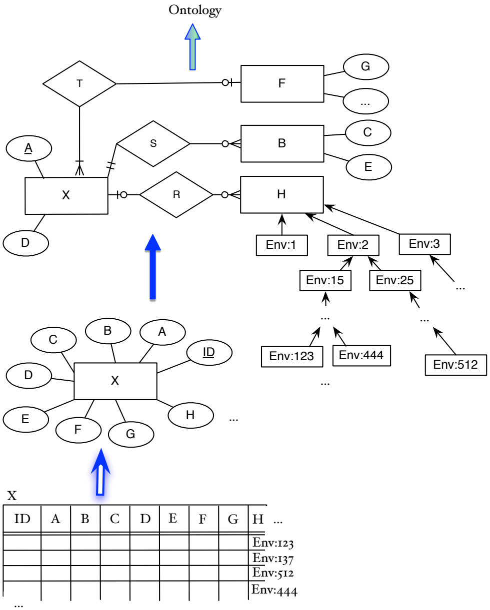
- Some data in the DB—mathematically instances—actually assumed to be concepts/universals/classes

General considerations for RDBMSs

- Some data in the DB—mathematically instances—actually assumed to be concepts/universals/classes
- ‘impedance mismatch’ DB values and ABox objects

General considerations for RDBMSs

- Some data in the DB—mathematically instances—actually assumed to be concepts/universals/classes
- ‘impedance mismatch’ DB values and ABox objects
- $\Rightarrow$
  values-but-actually-concepts-that-should-become-OWL-classes and values-that-should-become-OWL-instances


Ontology
T
F
G
...
S
B
C
E
A
X
R
H
D
Env:1
Env:2
Env:3
Env:15
Env:25
...
Env:123
Env:444
Env:512
C
B
A
ID
D
X
E
F
G
H
...
X
ID A B C D E F G H ...
Env:123
Env:137
Env:512
Env:444
...

General considerations for RDBMSs

- Reuse/reverse engineer the physical DB schema
- Reuse conceptual data model (in ER, EER, UML, ORM, ...)

General considerations for RDBMSs

- Reuse/reverse engineer the physical DB schema
- Reuse conceptual data model (in ER, EER, UML, ORM, ...)
- But,
    - Assumes there was a fully normalised conceptual data model,
    - Denormalization steps to flatten the database structure, which, if simply reverse engineered, ends up in the 'ontology' as a class with umpteen attributes
    - Minimal (if at all) automated reasoning with it

General considerations for RDBMSs

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

Manual Extraction

- Most database are not neat as assumed by 'Automatic Extraction of Ontologies' algorithms
- Then what?

Manual Extraction

- Most database are not neat as assumed by ‘Automatic Extraction of Ontologies’ algorithms
- Then what?
  - Reverse engineer the database to a conceptual data model
  - Choose an ontology language for your purpose

Manual Extraction

- Most database are not neat as assumed by ‘Automatic Extraction of Ontologies’ algorithms
- Then what?
  - Reverse engineer the database to a conceptual data model
  - Choose an ontology language for your purpose
- Examples:
  - Manual: Reverse engineering from DB to ORM model with, e.g., VisioModeler v3.1 or NORMA: the HGT-DB about horizontal gene transfer, ADOLENA for the portal for people with disabilities, EPnet with those amphorae
  - Automated: Lubyte & Tessaris's presentation of the DEXA'09 paper

Outline

Overview

- Thesauri galore in medicine, education, agriculture, ...
- Core notions of **BT** broader term, **NT** narrower term, and **RT** related term (and auxiliary ones UF/USE)
- E.g. the Educational Resources Information Center thesaurus:

```
reading ability
  BT ability
  RT reading
  RT perception
```

- E.g. AGROVOC of the FAO:

```
milk
  NT cow milk
  NT milk fat
```

- *How to go from this to an ontology?*

Problems

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

Simple Knowledge Organisation System(s): SKOS

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

See slides `SKOS.pdf`

A rules-as-you-go approach (1/2)

- 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 rules-as-you-go approach (2/2)

- Edit manually using an ontology editor:
  - make existing information more precise
  - add new information
  - automation of discovered patterns (rules-as-you-go); e.g.
    - observation: *cow* NT *cow milk* should become *cow* *<hasComponent>* *cow milk*
    – pattern: *animal* *<hasComponent>* *milk* (or, more generally *animal* *<hasComponent>* *body part*)
    — derive automatically: *goat* NT *goat milk* should become *goat* *<hasComponent>* *goat milk*
    other pattern examples, e.g., *plant* *<growsIn>* *soil type* and *geographical entity* *<spatiallyIncludedIn>* *geographical entity*

Outline

Natural language and ontologies

- Using ontologies to improve NLP; e.g.:
- Using NLP to develop ontologies (TBox)
- Using NLP to populate ontologies (ABox)
- Natural language generation from a logic

Natural language and ontologies

- Using ontologies to improve NLP; e.g.:
  - To enhance precision and recall of queries
  - To enhance dialogue systems
  - To sort literature results
- Using NLP to develop ontologies (TBox)
- Using NLP to populate ontologies (ABox)
- Natural language generation from a logic

Natural language and ontologies

- Using ontologies to improve NLP; e.g.:
  - To enhance precision and recall of queries
  - To enhance dialogue systems
  - To sort literature results
- Using NLP to develop ontologies (TBox)
  - Searching for candidate terms and relations: Ontology learning
- Using NLP to populate ontologies (ABox)

- Natural language generation from a logic

Natural language and ontologies

- Using ontologies to improve NLP; e.g.:
  - To enhance precision and recall of queries
  - To enhance dialogue systems
  - To sort literature results
- Using NLP to develop ontologies (TBox)
  - Searching for candidate terms and relations: Ontology learning
- Using NLP to populate ontologies (ABox)
  - Document retrieval enhanced by lexicalised ontologies
  - Biomedical text mining
- Natural language generation from a logic

Natural language and ontologies

- Using ontologies to improve NLP; e.g.:
  - To enhance precision and recall of queries
  - To enhance dialogue systems
  - To sort literature results
- Using NLP to develop ontologies (TBox)
  - Searching for candidate terms and relations: Ontology learning
- Using NLP to populate ontologies (ABox)
  - Document retrieval enhanced by lexicalised ontologies
  - Biomedical text mining
- Natural language generation from a logic
  - Ameliorating the knowledge acquisition bottleneck
  - Other purposes; e.g., e-learning (question generation), readable medical information

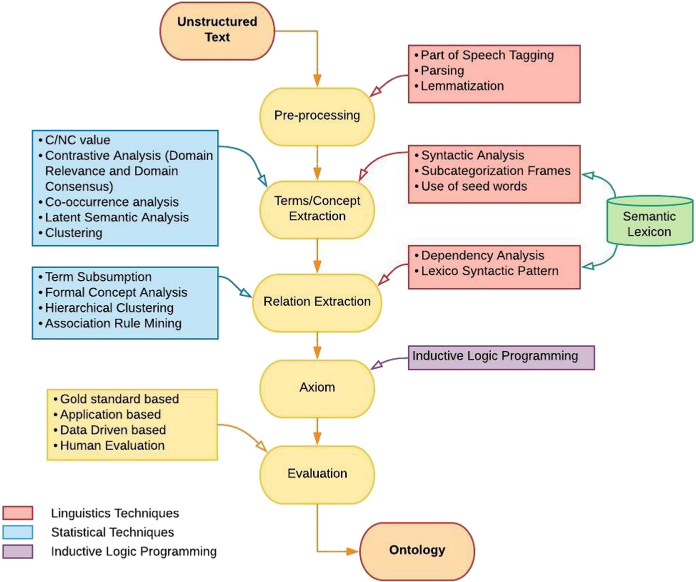
Examples (out of many)

- Generic tools: e.g.: for POS tagging, semantic tagging and annotation, ontology-based information extraction, morphological analysis etc. etc.
- Textpresso and similar tools
- Attempto Controlled English (ACE), rabbit, etc.; grammar engine, template-based approach

Background

- Ontology development is time consuming
- Bottom-up ontology development strategies, of which one is to use NLP
- We take a closer look at ontology learning limited to finding terms for a domain ontology

Sample pipeline



Bottom-up ontology development with NLP

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

Method to test automated term recognition

- Compare the terms of a manually constructed ontology with the terms obtained from text mining a suitable corpus
- Build an ontology manually
  - Lipoprotein metabolism (LMO), 223 classes with 623 synonyms
- Create a corpus
  - 3066 review article abstract from PubMed, obtained with a 'lipoprotein metabolism' search
- Automatic Term Recognition (ATR) tools, e.g.
  - Text2Onto: relative term frequency, TFIDF, entropy, WordNet, Hearst patterns
  - Termine: statistics of candidate term (total frequency of occurrence, frequency of term as part of other longer candidate terms, length)
  - 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

example figures for illustration: from Alexopoulou et al, 2008

What can go (went) wrong with some of the terms?

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

Ontology population: Typical NLP tasks

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

Requirements for NLP ontologies

- Domain ontology (at least a taxonomy)
- Text model, concerns with classes such as *sentence*, *text position* and locations like *abstract*, *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

Summary

1. RDBMSs
   - From conceptual model to ontology
   - From data to ontology
2. Thesauri
3. Natural language
   - Introduction
   - Ontology learning and population