Design: Ontology Engineering

Semantic Web (CSC751)

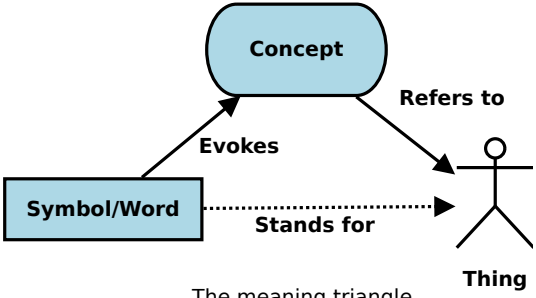
Ubbo Visser

Department of Computer Science
University of Miami

November 8, 2023

UNIVERSITY
OF MIAMI

Outline



The meaning triangle

Reading

- Ch. 8 [HKR09]

Final project presentation

- You have 10 minutes to present the project proposal to the class. Please stick to this time. Approximately 5 minutes for the feedback.
- Explain your project and describe how semantic technologies are used in your scenario.

Final project paper

- Submit the final project report by copying it to your home folder. The project report can be written in either LaTeX or Word using the same template given for the project proposal. All necessary sources to build the final PDF document need to be committed. Please describe (i) Motivation/Design, (ii) Tools used, API(s) and/or libraries used, and the programs you used/programmed in order to implement the solution. Include figures and tables where suitable. Think of this project report as writing a scientific paper.

Class presentation: things to consider

- Some form of visual presentation is effective.
- Effectiveness of explanations.
- Time management is a must.
- Correctness of content.
- Proficiency in answering questions.
- Experiment results, comparison results, evaluations, and potential application demonstrations.
- It is up to you to select the content.

Introduction

- Ontology engineering: building complex ontologies by providing methodologies and auxiliary tools for their development, evaluation, and maintenance.
- Methodologies may be subjective.
- Defining a ontology life cycle: steps for software development and maintenance.

Requirement analysis

- Is a semantic representation the better choice or not (relational databases)?
  - Non-semantic solutions already exist.
  - Added value with expensive modeling efforts.
- Ontology based systems:
  - Knowledge represented in semantic format can be easily exchanged as well as integrated with knowledge from other sources.
  - Implicit knowledge via a deduction algorithm.
- Tool support: specific tools, licenses, maturity, vendor support and interoperability.
- Which logic should be used? Large data and less expressive RDF(S), moderate and more expressive OWL 2 DL, and/or OWL 2 DL profiles.
- Domain, granularity, the tasks that the ontology needs to cover, and the expected and desired inferences.
- Degree of axiomatization: individuals, classes, and roles.

How?

- There is no uniquely correct way to build an ontology satisfying all the requirements.
- There exists best practices and modeling patterns.

Where?

- Human, unstructured, semi-structured, and structured sources of knowledge.
- Trying to define “game”, “democracy” or “school”.

Sources

- Human:
  - Domain expert. Can the domain expert formulate his/her knowledge in a logical way?
  - Knowledge engineer as a form of mediator to cast the knowledge into a logical specification.
  - "I know it when I see it".
  - Machine learning to learn rules based on positive/negative examples that the domain expert provides (decision trees, inductive logic programming, rule-based learning algorithms (RIPPER or Learning Classifier Systems), SVMs, formal concept analysis etc.).
- Unstructured:
  - All kind of textual resources (books, magazines, web pages etc.).
  - Exchanging formal specification from arbitrary text is an NP-hard problem
  - Parsing texts for part-of-speech tagging, named entity recognition, chunking, word-sense disambiguation, pronoun resolution, modern LLMs, ...
  - Transformation rule from parse trees to logic. e.g., `Ubbo`, `Sam` into individuals, `School` and intransitive verbs (`sleep`) into classes and transitive verbs (`like`) into roles. Some specific methods include:
    - Neural Semantic Parsers
    - Transfer Learning and Pre-trained Language Models (BERT, GPT)
    - Combinatory Categorial Grammar (CCG)
    - Supervised Learning with Annotated Corpora

Sources

- Unstructured
  - "Dish that contains fish" $\texttt{Dish} \sqcap \exists \texttt{contains.Fish}$
  - Normalization of words. e.g., nouns in nominative singular form and verbs in infinitive form.
  - Non-trivial to retain temporal information in standard ontology languages and no well-established best practices how to do this.
  - Common sense, or background knowledge. e.g., fish are animals (WordNet).
  - Control natural languages.
- Semi-structured:
  - Web pages having hyperlinks and wiki articles.
- Structured:
  - Directly accessible information.
  - Database to n-ary relationships via reification.
  - Ontology population from database rows.
  - Schema such as "every person has at least one nationality" into TBox axioms: $\mathit{Person} \sqsubseteq \exists \mathit{hasNationality}.\mathit{Nationality}$.
  - Upper-level ontologies (BFO, DOLCE, SUMO, Cyc, UFO).
  - Reuse other ontologies (ontology merging and ontology alignment).
  - Ontology mapping.

Ontology evaluation

- How is the quality of the ontology assessed?
  - Does it fulfill the intended purpose?
  - Does it infer the intended knowledge?
  - Do the logical inferences match with the reality?
  - Does the ontology and do the inferences help the user?
- Logical criteria:
  - Ontology is inconsistent or unsatisfiable, if it has no model. Then it is in almost every case a modeling error.
  - Existence of weaker version of class inconsistency ($C \sqsubseteq \bot$).

$$\begin{aligned} \mathit{Horse} &\sqsubseteq \neg \mathit{Flies} \\ \mathit{FlyingHorse} &\equiv \mathit{Horse} \sqcap \mathit{Flies} \end{aligned}$$

    Ontology is globally consistent, but `FlyingHorse` is inconsistent. The ontology becomes inconsistent when an individual of FlyingHorse is added.
  - Create classes only if individuals can be added.
  - Coherence: no inconsistent classes. A consistent ontology can be incoherent.
  - Less constraints, but consider disjointness, property restrictions etc.

Ontology evaluation

- Structural and formal criteria:
  - Taxonomic cycles.

$$\begin{aligned} \textit{Architecture} &\sqsubseteq \textit{Faculty} \\ \textit{University} &\sqsubseteq \textit{Building} \\ \textit{Faculty} &\sqsubseteq \textit{University} \\ \textit{Building} &\sqsubseteq \textit{Architecture} \end{aligned}$$

    This ontology collapses with all equivalent concepts.
  - Rigidity: e.g., person cannot stop being person, where as a student can stop being a student while retaining his/her existence and attributes.
  - Non-rigidity, identity, unity, and dependence.
- Accuracy criteria:
  - Ontology accurately captures the domain.
  - Judgment of humans are subjective.

What must be avoided

- Don't forget disjointness:

$Man \sqsubseteq Human$	$Human \sqsubseteq Man \sqcap Woman$	$Woman \sqsubseteq Human$
$Man(Jack)$		$Woman(Jill)$

This knowledge base does not entail that $\neg Woman(Jack)$. This is because no logical reasons prevent *Jack* being both *Man* and *Woman*. In order to fix this you need to include $Man \sqcap Woman \sqsubseteq \bot$.

Explicitly consider all siblings, i.e., classes having a common superclass, whether it is possible that an individual is an instance of both classes. If not, declare them as disjoint.

What must be avoided

- Don't forget role characteristics:

  Consider for every role occurring in an ontology whether it represents relations: transitive, symmetric, functional, inverse functional, reflexive, irreflexive, antisymmetry, role disjointness, and interdependencies involving role chains.

- Don't choose too specific domains or ranges:

  If the domain and range are set, make sure that the class expressions are disjoint. If not there will be uncontrollable inferences.

What must be avoided

- Be careful with quantifiers:
  - When translating from natural language statements into logical axioms, existential quantification is more natural. E.g., birds have wings $\texttt{Bird} \sqsubseteq \exists \texttt{has.Wing}$. But the erroneous translation
    $\texttt{Bird} \sqsubseteq \forall \texttt{has.Wing}$ says that birds have only wings or no wings at all.
  - Use universal quantification with words such as "only", "exclusively" or "nothing but".
  - $\texttt{Happy} \equiv \forall \texttt{hasChild.Happy}$ and
    $\texttt{Happy} \equiv \forall \texttt{hasChild.Happy} \sqcap \exists \texttt{hasChild.Happy}$

  Make sure that the intended meaning is correctly casted into role quantification. Use existential quantification as default. Be aware that universal quantification alone does not enforce the existence of a respective role.

What must be avoided

- Don't mistake parts for subclasses:

$Finger \sqsubseteq Hand$	$Hand \sqsubseteq Arm$	$Arm \sqsubseteq Body$
$Toe \sqsubseteq Foot$	$Foot \sqsubseteq Leg$	$Leg \sqsubseteq Body$
	$Arm \sqcap Leg \sqsubseteq \bot$	

  - This knowledge base infers that $Hand \sqsubseteq Body$. Is this natural?
  - If we do this, we are mistaken meronymy for hyponymy, i.e., part-of relations for subclasses relations.
  - Meronymy, i.e, the semantic relation that holds between a part and the whole, is modeled by a dedicated transitive role `partOf` and $\texttt{partOf}^-$.

$Finger \sqsubseteq \exists partOf.Hand$	$Hand \sqsubseteq \exists partOf.Arm$	$Arm \sqsubseteq \exists partOf.Body$
$Toe \sqsubseteq \exists partOf.Foot$	$Foot \sqsubseteq \exists partOf.Leg$	$Leg \sqsubseteq \exists partOf.Body$
	$Arm \sqcap Leg \sqsubseteq \bot$	

A class `A` should be modeled as a subclass of `B` only if the statement "every `A` is a `B`" makes sense and is correct.

What must be avoided

- Watch the direction of roles:
  - For nouns, instead of `author`, use `authorOf` or `hasAuthor`. For verbs, instead of `wrote` or `written`, use `writtenBy`.

  When introducing a new property or role names, add a comment that clarifies what its source and target are. Moreover, use names which allow only one unique intuitive reading.
- Don't confuse class subsumption and class equivalence:
  - $\texttt{Orphan} \equiv \texttt{Person} \sqcap \forall \texttt{hasParent}.\neg\texttt{Alive}$

  Only if a class description is both necessary and sufficient, an equivalent statement should be used.
- Don't translate too verbally:
  - "University staff members and students get a login". $\texttt{A} \sqcap \texttt{B} \sqsubseteq \exists \texttt{gets}.\texttt{Login}$ vs. $\texttt{A} \sqcup \texttt{B} \sqsubseteq \exists \texttt{gets}.\texttt{Login}$

  If in doubt about the correct formalization, two strategies that might help are paraphrasing and testing.

Modules

- Import and reuse.
- How much do we import: coverage vs. economy.
- Coverage should preserve entailments, i.e., $\mathcal{O} \cup \mathcal{E} \models A \sqsubseteq B$ then $\mathcal{O} \cup \mathcal{E}' \models A \sqsubseteq B$.
- OWL API is a highly recommended module extractor.

Acknowledgement

Acknowledgement

The slides for this course have been prepared by Saminda Abeyruwan.


Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph.
Foundations of Semantic Web Technologies.
Chapman & Hall/CRC, 2009.