Seminar on

Ontology Integration and Evolution

Hoang Huu Hanh
14.12.2004
Content

- Introduction
- Ontology Normalization and Modularization
- Ontology Integration
- Ontology Integration Tools
- Ontology Evolution
- Conclusions
Ontology Normalization & Modularization

- Ontology normalization and modularization untangle an ontology into multiple independent disjoint trees
- Why we need the Normalization and Modularization?
  - To facilitate easy ontology integration
  - A way to modularize and normalize ontologies needs to be defined
- Ontologies must be modularized in order to simplify the ontology integration
- It is useful to normalize and modularize ontologies to support re-use, maintainability and evolution
Criteria for normalization
- The basis of the normalization of an ontology: its primitive skeleton
- The primitive skeleton is a tree formed by subsumption relationship of the primitive concepts of the ontology
- A primitive concept is defined by necessary conditions

Certain rules hold for the primitive skeleton:
- Branches of the skeleton must form trees
- Branches of the skeleton must be homogenous
- The primitive skeleton should distinguish between self standing and partitioning/refining concepts.
Ontology Modularization

- Example:

```
  Holiday
     /   \
   Adventure    Relaxation
       /    \
    Alpine Climbing  Diving
           /   \
         Beginner  Advanced
                      /  \
                      Expert
```
Ontology Normalization & Modularization (4)

- Ontology Modularization

Diagram:
- Activity
  - Diving
  - Climbing
- Holiday
  - Adventure
  - Relaxation
- Skill
  - Beginner
  - Advanced
  - Expert
- Location
Ontology Integration

Why integrating?

- Number of ontologies in the same area:
  cover different aspects, contain overlapping information
- Use of multiple ontologies
  • e.g. public + company
- Bottom-up creation of ontologies
  • e.g. merge of companies can lead to a merge of their ontologies
Ontology Integration

Ontology_A

Library
  part_of
  People
    {name, uid, address, type}
    Periodical-Publication
      {title, editor, pages, periodicity}
        Journal
          {area, issue_no}
          Newspaper
            {date}

Ontology_B (preferred)

Library
  part_of
  Person
    {name, id, address, role}
    Publication
      {title, editors, pages}
      Book
        {author, ISBN, style}
        Newspaper
          {date}

Merged_ontology

Library
  part_of
  Person
    {name, id, address, role}
    Publication
      {title, editors, pages}
      Periodical-Publication
        {periodicity}
        Book
          {author, ISBN, style}
        Newspaper
          {date}

Magazine
  {category, issueNo, periodicity}

Institute for Software Technology and Interactive Systems
Ontology Integration

Terminology (1): Mapping / Aligning

- **Mapping**
  Relating similar concepts or relations from different sources to each other by an equivalence relation

- **Aligning**
  Bring two or more Ontologies into mutual agreement making them consistent and coherent

(Klein, 2001)
Terminology (2): Merging / Integrating

- **Merging / Integrating**
  Creating a new Ontology from two or more existing ontologies with *overlapping* parts

- **Merging**
  Merging different ontologies about the same subject into a single one that "unifies" all of them.

- **Integrating**
  Building a new ontology reusing other available ontologies (assemble, extend, specialize).

D: domain; O: ontology

(Klein, 2001 & Pinto et al, 1999)
Ontology Integration

Terminology (3)

- **Combining**: Using two or more different Ontologies for a task in which their mutual relation is relevant
- **Translating**: Changing the representation formalism of an ontology while preserving the semantics
- **Transforming**: Changing the semantics of an ontology slightly to make it suitable for purposes other than the original one.
Mismatches between Ontologies

- Language level mismatches
  - Mismatches between the mechanisms to define classes, relations

- Ontology or Model level mismatches
  - Mismatch is a difference in the way the domain is modeled
Ontology Integration

Language level mismatches
Ontologies written in different ontology languages

- **Syntax**: different Ontologies languages use different syntaxes
  - e.g. `<rdfs:Class ID = “Chair”>` in RDF Schema vs. `(defconcept Chair)` in LOOM
  - solution: rewrite mechanisms

- **Logical representation**: different languages construct – logically equivalent statements
  - e.g. disjoint A B vs.
    - A subclass-of (not B), B subclass-of (not A)
  - solution: translation rules

- **Semantics of primitives**: same name, different interpretations
  - e.g. interpretations of `<rdfs:domain>`
    - OIL RDF Schema: *intersection of the arguments*
    - RDF Schema: *union of the arguments*

- **Language expressivity**: some languages are able to represent things that are not expressible in other languages
  - e.g. negation
Ontology or Model-level mismatches

Two types of mismatches at the ontology level:

- **Conceptualization mismatch**: difference in the way a domain is interpreted.
  - Results in different ontological concepts, different relations between those concepts
  - Cannot be solved automatically but require knowledge and decision of a domain expert.

- **Explication mismatch**: Difference in the way the conceptualization is specified.
Ontology Integration

Conceptualization mismatch
difference in the way a domain is interpreted

● **Scope**
  Two classes seem to represent the same concept but do not have exactly the same instances, although they intersect
  ▪ e.g. Class “employee” several administrations use slightly different concepts of employee

● **Model coverage and granularity**
  Mismatch in part of the domain that is covered by the ontology, or the level of detail to which that domain is modeled.
  ▪ example:
    • Ontology of cars but not trucks.
    • Ontology that classify trucks into a few categories
    • Ontology that makes fine-grained distinctions between types of trucks
Explication mismatches
difference in the way the conceptualized is specified

- **Style of modeling**
  - **Modeling style paradigm**
    different paradigms can be used to represent concepts such as time, action, plans, causality.
    - e.g. different temporal representations based on interval logic vs. based on point.

  - **Concept description / modeling conventions**
    e.g. Distinction between two classes can be modeled using a qualifying attribute or by using a separate class

```
Ellipse          Circle
  
Circle          Ellipse
```

“Bowtie” Inconsistency
Ontology Integration

Explication mismatches (cont.)

- **Terminological Mismatches**
  - **Synonym terms / term mismatch**
    Concepts are represented by different names
    - e.g. *car* vs. *automobile*
    - e.g. different natural languages
    - solution: use of thesauri
  - **Homonym terms / concept mismatch**
    Meaning of the term is different in another context.
    - e.g. "conductor" in music domain vs. in electric engineering domain
    - Hard to handle → human knowledge is required
  - **Encoding**
    different formats
    - e.g. date *dd-mm-yyyy* vs. *mm/dd/yyyy*
    - e.g. *miles* vs. *kilometres*
    - Solution: transformation steps / wrapper.
Ontology Integration

Similarity measures

- Linguistic / syntactic similarity
  - synonym lists
  - string matching: shared substrings, common prefixes, common suffixes

- Semantic similarity
  - structure of the concept
  - concept position in the ontology
  - graph-based analysis

- Specific to an application area
Ontology Integration

Integration Process

- Find the places where ontologies overlap
- Relate concepts that are semantically close by equivalence and subsumption relations (aligning)
- Check the consistency, coherency and non-redundancy of the result

(D. McGuinness, 2000)
Ontology Integration

Need for tools

● Ontology integration is a complicated process mostly done by hand
  ▪ it is difficult to find the terms that need to be aligned
  ▪ the consequences of a specific mapping (unforeseen implications) are difficult to see

● Semi-automatic tools
  ▪ guide the user through the process and focus his attention on the likely points for action
  ▪ enable reusability of alignments in the context of ontology maintenance
## Ontology Integration Tools Overview

<table>
<thead>
<tr>
<th>Issues</th>
<th>PROMPT</th>
<th>Chimaera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language level mismatches</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Representation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expressivity</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ontology level mismatches</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paradigm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concept description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coverage of model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scope of concepts</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>Synonyms</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>Homonyms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encoding</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Practical problems</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finding alignments</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>Diagnosis of results</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Repeatability</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

*A: Automatic solution; U: suggestion to the user*  
*(Klein, 2001)*
Protégé with PROMPT

- Tool for interactive ontology-merging in Protégé
  (Stanford Medical Informatics at Stanford University School of Medicine)

- PROMPT approach *is*:
  - Partial automation
  - Algorithms based on
    - concept-representation structure
    - relations between concepts
    - user’s actions

- PROMPT approach *is not*:
  - Complete automation
  - Algorithm for matching concept names
The PROMPT Algorithm

- Make initial suggestions
- Select the next operation
- Perform automatic updates
  - Find conflicts
  - Make suggestions
Example: *merge-classes*

- **Agency** employee
- **Employee**
- **Agent**
- **Customer**
- **Traveler**

- **Agent** subclass of **Employee**
- **Employee** subclass of **Agent**
- **Agency employee** subclass of **Agent**
- **Agent** has client **Customer**
- **Agent** has client **Traveler**
Example: \textit{merge-classes (II)}
Analyzing Global Properties Locally

- **Global properties**
  - classes that have the same sets of slots
  - classes that refer to the same set of classes
  - slots that are attached to the same classes

- **Local context**
  - incremental analysis
  - consider only the concepts that were affected by the last operation
After a User Performs an Operation

- For each operation
  - perform the operation
  - consider possible conflicts
    - identify conflicts
    - propose solutions
  - analyze local context
  - create new suggestions
  - reinforce or downgrade existing suggestions
Operations and Conflicts

- Different operations
  - merge, copy: classes, slots, instances (deep vs. shallow)
  - remove parent

- Different conflicts
  - name conflicts
    more than one frame with the same name
  - dangling references
    reference to non existing frame
  - redundancy in the class hierarchy
    more than one path from a class to a parent other than root
  - slot-value restrictions that violate class inheritance
Institute for Software Technology and Interactive Systems
The PROMPT Tool Features

- Setting a preferred ontology
  - e.g. stable ontology to which changes are discouraged
- Maintaining the user’s focus
  - first the items that include frames related to the latest operations
- Providing feedback to the user
- Preserving original relations
  - subclass-superclass relations
  - slot attachment
  - facet values
- Linking to the direct-manipulation ontology editor
- Logging and reapplying the operations
Anchor-PROMPT: Using Non-Local Contexts

- **Input:**
  - A set of anchor pairs

- **Output:**
  - A set of related terms with similarity scores

- **Where do anchors come from?**
  - Lexical matching
  - Interactive tools
  - User-specified
Generating Paths in the Graph

Design-a-Trial, S.Modgil, et.al.; CMT, I.Sim et.al.
Similarity Score

- Generate a set of all paths (of length < L)
- Generate a set of all possible pairs of paths of equal length
- For each pair of paths and for each pair of nodes in the identical positions in the paths, increment the similarity score
- Combine the similarity score for all the paths
PROMPT Evaluation

Relative to

- Human experts performance
  - Humans followed 90% of Prompt’s suggestions for ontologies merging.
  - Humans followed 75% of the conflict-resolution strategies that PROMPT proposed.

- Other ontology-merging tools
  - PROMPT vs. Generic Protégé
    60 operations using Generic Protégé, 16 operations using PROMPT
  - PROMPT vs. Chimaera
    30% more correct suggestions than Chimaera

(N. Noy and M. Musen, 2000)
Ontology Evolution

- Information on the Web is continuously changing!
  - however, synchronization can not be enforced
- Humans used to live with those problems:
  - heuristics and background knowledge to filter outdated and wrong information
  - exponential growth reduces problems
- At the Semantic Web computers will use data!
  - invalid semantics will make all reasoning useless!
- Current SW work silently assumes stability:
  - awareness of problems, but only a few efforts
Ontology Evolution: Causes of Change

- Ontology: formal specification of a shared conceptualization of a domain
- Domain change:
  - change in real world, e.g. merge of two departments
  - well known from database schema versioning
- Change in the conceptualization:
  - how the world is perceived, e.g. better understanding
  - may also be caused by adaptation for other task
- Specification change:
  - other formalism, i.e. a translation
Ontology Evolution: Effects of Change

- **Instance data** that conforms to the ontology
- **Other ontologies** that are built from, or import the ontology
- **Applications** that use the ontology
Ontology Versioning

- The ability to manage ontology changes and their effects by creating and maintaining different variants of the ontology

- Versioning methodology should:
  - provide an unambiguous reference to the intended definition for every use of a concept or a relation; (identification)
  - make the relation of one version of a concept or relation to other versions of that construct explicit; (change tracking)
  - provide methods to give a valid interpretation to as much data as possible (transparent evolution)
Summary

- Ontology integration is a complicated process
  - find overlap; relate concepts; check consistency
- Mismatches
  - language level
  - ontology level
- Similarity measures
- Ontology merging tools
  - require interaction with user
  - identify overlaps, suggest actions to be taken and perform consistency checks
  - focus on concepts, slots, is-a relations; matching instances, facets, axioms is a future work
- Ontology versioning
  - important in an open environment such as We
  - more comprehensive schemes for interoperability of ontologies are required.
Resources

Thank you for your attention!