Semantic Interoperability & Knowledge Integration: Ontologies

CS 431 – March 14, 2007

Carl Lagoze – Cornell University

Web Information Systems

- This course is about building systems that interoperate over heterogeneous information distributed over the web.

Observations on interoperability so far

- Model interoperability
  - FRBR
- Identity interoperability
  - DOIs
  - URI
- Protocol interoperability
  - HTTP
- Architecture interoperability
  - The Web
- Syntactic interoperability
  - XML
- Vocabulary interoperability
  - XML Schema

But this is not enough…

- To build effective distributed information systems which can be interpreted by humans and machines we need the ability to exchanges messages about concepts and entities. We use words and identifiers to talk about the world. So we need some way of establishing agreement about the words and identifiers. This is accomplished with an ontology.

Example: Book-Shopping Bot

- Find the best deal on a book from various suppliers considering price, shipping, tax, availability, reputation, etc.
- Terminologies for everything else is variable
- ISBN is a good starting point for an identity!

Example: Medical Bot

- Question: What is the best drug to treat my problem?
- This problem has lots of facets
  - Anatomy
  - Diagnosis
  - Symptoms
  - Economics
  - Occupation
  - Age
  - Geographic locatoin

Note: Why not just an RDBMS and views

- Common schema are hard to achieve
- High variance among:
  - Formats (domain, scale, precision)
  - Names
  - Structure
  - Presence or absence of data
  - Constraints

This is not just about merging vocabularies

- Words and concepts have lots of relationships
- Think about some:
  - is-a (subsumption): a horse is a mammal
  - Part/whole: a horse has a tail
  - Connection/assocation: a garage is connected with a house
  - Dependence: a child must have a biological mother
  - Equivalence: Cornell University and EIN 15-0532082 are the same
- These relationships allow inferences about knowledge

What is an ontology?

- Many definitions from different domains
  - Philosophy – A Systematic Account of Existence
  - A.I. - An explicit specification of a conceptualization (the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them) *Gruber 1993*
- Our context – web semantic interoperability
  - Formal explicit description in a domain of discourse (**classes, concepts**)
  - Attributes of concepts (**slots**, **properties, relationships**)
  - Slot restrictions (**facets**)
- A **Knowledge Base** is an ontology combined with instance data

Two ways that ontologies are applied

- Bottom-up integration:
  - parties agree on ontology
  - Resolve internal views to ontology
  - Used in B2B e-commerce and electronic data interchange (EDI)
- Top-down integration:
  - Ontology providers the framework for analyzing/classifying information from distributed parties
  - Machine learning/data mining

Knowledge Integration: So what do we need?

- Ontology meta-model
- Encoding scheme for ontologies
- Domain specific ontologies
- Encoding scheme for instances
- Query mechanism
- Reasoning and inferencing

Overview of Ontology and Knowledge Base Development

- Define the classes
- Arrange the classes into a taxonomic hierarchy
  - establish class/sub-class relationships
- Define slots and their restrictions
- Define instances

Why Ontologies (1)?

- Sharing a formalized definition of information structure among people or software
  - e.g., ShopBots extracting and aggregating information from different sites
  - formalization of notation and decidability is important

Why Ontologies (2)?

- Enable reuse of domain knowledge
  - modularize development process
  - e.g., share common concepts of time (events, situations) in domain specific ontologies

Why Ontologies(3)?

- Separate operational from domain knowledge
  - avoid hard-coding domain knowledge into programs
  - parameterize code to allow use in different domains
  - allow easy modification of domain knowledge without code changes

Some guiding rules of ontology design

- In most cases there are many ways to model a domain
- Ontology development, like program development, is by nature iterative
- The ontology should closely correspond to the objects (nouns) and relationships (verbs) in the sentences describing your domain of interest

Ontology Development (1)

- Define the scope
  - What domain does it describe?
  - What applications will be built upon it?
  - What are the questions for which it should provide answers?
    - *competency questions* that serve as tests of ontology.
  - Who are its users and maintainers?
  - Limiting the scope is vital to a usable ontology.
    - Don't include extraneous information!

Ontology Development (2)

- Search available online ontologies and determine utility of them.
  - http://www.daml.org/ontologies/
  - http://protege.stanford.edu/plugins/owl/owl-library/
- Increases possibility of interoperability with other applications

Ontology Development (3)

- Enumerate important terms in ontology
  - Concepts and properties
  - Ignore relationships for now, just brainstorm
- Establish a naming convention
  - capitalization
  - use of delimiters
  - singular or plural
  - prefixes

Ontology Development (4)

- Define concepts and concept hierarchy
  - Top-down
  - Bottom-up
  - Remember transitivity of class hierarchy
  - Depth and breadth issues
    - Avoid single sub-class
    - Excessive # of siblings (> 12) indicates possible need for new sub-classing

Ontology Development (5)

- Define slots or properties of classes
  - data properties
    - names
    - flavors
    - colors
  - object properties
    - whole/part relationships
    - other semantic relationships among individuals
  - Reflect class/sub-class hierarchy
    - Slots should distinguish sub-classes
    - Attach slot at most general point in hierarchy
    - Remember that all sub-classes inherit slot

Ontology Development (6)

- Define facets of slots
  - Data type of data slots
  - Domain and range of object slots
    - Again obey class generality rule
  - Slot cardinality

Ontology Development (7)

- Test with instances

Issues (1) – Multiple Inheritance

- Most systems allow it
- Frequently necessary to model a domain
- Make sure slot inheritance works

Issues (2) – Classes vs. Slots

- E.g., wine with slot color, or sub-classes for red, white, rose
- If classes with different slot values become restrictions for other slots in other classes, create a new class for distinction
  - example – consider car color vs. wine color

Issues (3) – Instance or Class?

- Answer is domain specific and application specific
  - Magnet Pinot Noir vs. Magnet Pinot Noir 2003
- Remember that instances are essentially the leaves in the knowledge base hierarchy
  - no notion of sub-instance
- Instances should be answers to competency questions

More Issues

- Disjoint classes
  - Can't have any instances in common
  - Pay attention to open world issues
- Inverse slots
  - Usually unnecessary to represent
    - system can infer information
    - "reverse queries" are possible
  - Sometime useful for understanding
    - system provide way of automatically completing

Ontology Tool

- Protégé
  - http://protege.stanford.edu/
- Open Source, Java Based
- Export to a variety of formats