

Tutorial - 1

How to build an ontology

Next Generation Web Group
Universität Innsbruck

Outline

- Main components of ontology
- well-known ontologies
- how to build ontologies
- build on by yourself - Protege

Why to build an ontology?

- To share common understanding of the structure of information among people or software agents
- to enable reuse of domain knowledge
- to make domain assumptions explicit
- to separate domain knowledge from operational knowledge
- to analyze domain knowledge

Main components of an Ontology

- Five kinds of components:
 - classes:
 - concepts of the domain or tasks, which are usually organized in taxonomies
 - in univ-ontology: student and professor are two classes
 - relations:
 - a type of interaction between concepts of the domain
 - such as: subclass-of, is-a

Main components of an Ontology (Cont.)

- Five kinds of components:
 - functions:
 - a special case of relations in which the n-th element of the relationship is unique for the n-1 preceding elements
 - Such as: Price-of-a-used-car can define the calculation of the price of the second-hand car on the car-model, manufacturing data and kilometres
 - axioms
 - model sentences that are always true
 - such as: if the student attends both A and B course, then he or she must be a second year student
 - instances
 - to represent specific elements
 - such as: Student called Peter is the instance of Student class

Examples

- Give you idea on how ontology look likes and how it works
 - **Ontology libraries in DAML (www.daml.org)**
 - Top level ontology: Standard Upper Ontology
 - Linguistic ontology: WordNet
 - General Ontology: Cyc, UNSPSC, ecl@ss
 - Domain ontology: CHEMICALS, UMLS
 - Research ontology: KA2

Building an ontology means

- Defining classes in the ontology
- arranging the classes in a taxonomic (subclass - superclass) hierarchy
- defining slots and describing allowed values for these slots
- filling in the values for slots and for instances

Step-by-Step methodology

- Step 1: Determine the domain and scope of the ontology
 - what is the domain that the ontology will cover
 - for what we are going to use the ontology
 - for what types of questions the information in the ontology should provide answers
 - who will use and maintain the ontology
- Example: University Ontology

Step 2

- Step 2: consider reusing existing ontologies
 - Ontolingua ontology library (<http://www.ksl.stanford.edu/software/ontolingua/>)
 - the DAML ontology library (<http://www.daml.org/ontologies/>).
 - publicly available commercial ontologies (e.g., UNSPSC (www.unspsc.org), RosettaNet (www.rosettanet.org), DMOZ (www.dmoz.org))
- Example: University Ontology

Step 3

- Step 3: Enumerate important terms in the ontology
 - a list of terms which are important to explain to the user
 - what properties do those terms have?
 - What would you like to say about those terms
- Important: to get the comprehensive list of terms
- Example: University Ontology

Step 4

- Step 4: Define the classes and the class hierarchy
 - **Top-down**
 - starts with the most general concepts
 - subsequent specialization of the concepts
 - **bottom-up**
 - starts with the most specific concepts or classes, the leaves of the hierarchy
 - subsequent grouping of these classes into more general concepts
 - **Middle-out**
 - combination of the top-down and bottom-up approaches
 - starts with the salient concepts first
 - generalize and specialize them appropriately
 - **Subclass relation**
 - if class B represents a concept that is a "kind of" A, then B is subclass of A
- Example: University Ontology

Step 5

- Step 5: Define the properties of classes - slots
 - define the internal structure of concepts
 - several types of properties that can become slots in an ontology
 - intrinsic properties: flavor of a wine
 - extrinsic properties: a wine's name
 - parts: physical and abstract "parts" (e.g, the courses of a meal)
 - relationships to
 - other individuals (e.g, the maker of the wine)
 - All subclasses of a class inherit the slot of that class.
- Example: University Ontology

Step 6

- Step 6: Define the facets of the slots
 - value type,
 - allowed values,
 - the number of the values (cardinality)

Step 6 - Cont.

- Step 6: Define the facets of the slots
 - Several common facets
 - slot cardinality:
 - how many values a slot can have
 - single cardinality and multiple cardinality
 - minimum and maximum cardinality
 - Slot-value type
 - string, number, boolean, enumerated, instance-type slots
 - Domain and range of a slot
 - domain
 - range

Step 6 - Cont.

- Basic rules for determining a domain and a range of a slot
 - find the most general classes, while don't use the two generic classes
 - subclass + super class ==> super class
 - subclass + subclass ==> super class

Step 7

- Step 7: create instances
 - defining an individual instance of a class requires:
 - choosing a class
 - creating an individual instance of that class
 - filling in the slot values

Defining classes and a class hierarchy

- There is no single correct class hierarchy for any given domain.
- The hierarchy depends on the possible uses of the ontology
- The level of detail is depend on the applications

Guideline for defining class hierarchy

- Ensuring that the class hierarchy is correct
 - is-a, kind-of relation
 - Mistake: include both a singular and plural version of the same concept in the hierarchy making the former a subclass of the latter
 - Transitivity of the hierarchical relations
 - evolution of a class hierarchy
 - classes and their names: classes represent concepts in the domain and their names just denote these concepts
 - Mistake: synonyms for the same concept do not represent different classes
 - avoiding class cycles

Guideline for defining class hierarchy - Cont.

- Analyzing siblings in a class hierarchy
 - siblings are the classes that are direct subclasses of the same class
 - all the siblings must be at the same level of generality
 - there is no hard rules for the number of direct subclasses that a class should have (around 2-12)
 - if a class has only one direct subclass that maybe a modeling problem for the ontology is not complete
 - if there are more than a dozen subclasses for a given class then additional intermediate categories may be necessary.

Guideline for defining class hierarchy - Cont.

- Multiple inheritance
 - a class can be a subclass of several classes
- when to introduce a new class (or not)
 - not easy
 - thumb rules
 - subclass has additional property that the superclass does not have
 - subclass has different restrictions from those of the superclass
 - subclass participates in different relationships than the superclasses
 - classes in the hierarchy do not have to introduce new properties
 - we should not create subclasses of a class for each additional restriction

Guideline for defining class hierarchy - Cont.

- A new class or a property value?
 - Need to decide whether to model a specific distinction (such as color of the wine) as a property value or as a set of classes, this depends on the scope of the domain and the task at hand
 - If the concepts with different slot values become restrictions for different slots in other classes, otherwise, we represent the distinction in a slot value.
 - If a distinction is important in the domain and we think of the objects with different values for the distinctions as different kinds of objects, then we should create a new class for the distinction
 - A class to which an individual instance belongs should not change often

Guideline for defining class hierarchy - Cont.

- An instance or a class?
 - Depends on the applications of the ontology
 - starts with deciding what is the lowest level of granularity in the representation, the level of granularity depend on the application of the ontology
 - individual instances are the most specific concepts represented in a knowledge base

Guideline for defining class hierarchy - Cont.

- limiting the scope
 - the ontology should not contain all the possible information about the domain: you do not need to specialize (or generalize) more than you need for your application (at most one extra level each way).
 - The ontology should not contain all the possible properties of and distinctions among classes in the hierarchy
- Disjoint subclasses
 - classes are disjoint if they cannot have any instance in common
 - specifying that classes are disjoint enables the system to validate the ontology better

Defining properties

- Inverse slots
 - a value of slot may depend on a value of another slot
 - made, made-by
 - reduce the redundancy,
 - insure the consistency of the knowledge base
- Default values

Naming convention

- Defining naming conventions for concepts in an ontology and then strictly adhering to these conventions not only makes the ontology easier to understand but also helps avoid some common modeling mistakes
 - We need to define a naming convention for classes and slots and adhere to it

Naming convention

- Some features of naming convention
 - does the system have the same name space for classes, slots and instances?
 - Does the system allow having a class and a slot with the same name
 - is the system case-sensitive?
 - What delimiters does the system allow in the names
 - can name contain spaces, commas, asterisks, and so on.

Naming convention

- Capitalization and delimiters
 - improve the readability of an ontology
- Singular or plural
 - should be consistent
- Prefix and suffix conventions
- Other naming considerations
 - avoid abbreviations

Homework

- Building your own University Ontology based on this tutorial
 - follow the each step and answer the question mention in each step
 - build in the Protégé tool
- Present your work (presentation around 3-5 slides) during next Tutorial - Week 3