CS 5100: Foundations of Artificial Intelligence

Ontology Design & Development

Prof. Amy Sliva
October 20, 2011

Outline

- Projects and grading
- Midterm!?!
- Ontology design
- Assignment 4

Comments on Assignment 2

- Major challenge
  - Design data structure and objects that makes the program efficient
- Minor challenge
  - Make the output nice for humans to read/understand

- Major mistake(s):
  - Reading the KB file each time PLFC is called
  - Reading the KB file each time a percept is processed!!!
  - Changing the KB file to effect adding knowledge!!!!!!!

Comments on Assignment 2

- Best approach
  - Read KB.txt once at "top level" (or with a method call in a class) and create nice data structures
  - For each percept, follow the forward chaining algorithm

- Data structures
  - **Rule class with conclusion and premises fields**
  - List of known facts (or fact objects) for quick checking—define a function factKnown(p)
  - Dictionary where each premise indexes a list of rules containing that premise

Grading Assignment 2

- 40 points for written problem set
- 60 points for programming
  - 30 points for correct program (forward chaining and output)
  - 12 points for data structure/OO solution
  - 12 points for readability of output and source code
  - 6 points for following project specification
    e.g., PLFC() is to take no arguments, required to use OO, etc.
  - 10 points extra credit for making facts in the KB explicit

Grading Assignment 3

- 40 points for written problem set
- 60 points for programming
  - 30 points for correctness
  - 12 points for program design
  - 12 points for readability of output and source code
  - 6 points for conforming to specification

Midterm—October 27!

- **Intelligent agents**
  - Definitions, structure, etc.
- **Logic and reasoning (all prior homework)**
  - Syntax & semantics, unification, conversion to CNF, forward and backward chaining, resolution
- **State space search and planning model for problem solving**
- **Search algorithms**
  - Performance tradeoffs
  - Heuristics
- **Practice planning problem**
  - STRIPS/PDDL planning operators (components, how used)
  - Planning graphs (don't worry, you won't have to draw one)
  - Situation calculus
- **Ontology design**

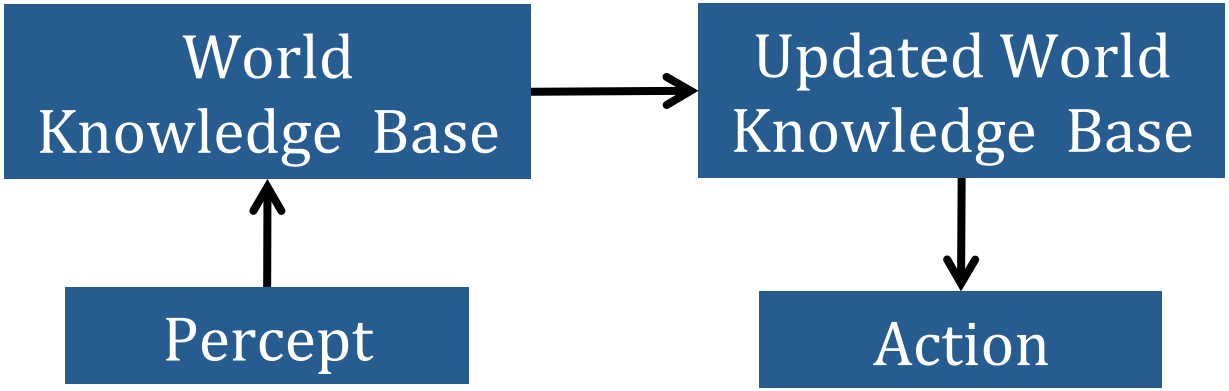
Practice problem (from prior midterm)

Apply Situation Calculus with logical deduction (by resolution) to solve part of a planning problem where the goal is that you are in your car.

The initial state: you are in your house, and the car is in the driveway. The goal is that you are in your car. In order for that to happen, you have to be at the location where your car is, and you have to get into your car.

Define logical constants, functions, and predicates to represent the domain. Write down the general world-knowledge axioms and some logical sentences describing the initial state S0. Show how you would derive the answer, and be sure to clearly indicate how the resulting plan be represented in logic.

Review: elements of agent knowledge



- Three main components

✔1. A formal language for expressing knowledge declaratively
   - **Knowledge representation** **Logic!**

2. A knowledge base design to express what is known
   - **Knowledge engineering** or **ontology**

✔3. Algorithms to use and update the knowledge base
   - **Automated inference** **Forward and backward chaining!**

Ontology design

- Ontology—design of formal **conceptual structures**
  - Knowledge representation of **real world** concepts
- Use data structures to put ontologies in an agent/software
  - Implement programs to manipulate and reason about them
- What's the point?
  1. Share common understanding of structure of information
  2. Enable reuse of domain knowledge
  3. Make domain assumptions explicit
  4. Separate domain knowledge from operational knowledge
  5. Analyze domain knowledge

Major elements of ontologies

- **Taxonomy**—"isa" hierarchy
  - Class and subclass hierarchy
- **Partonomy**—"is-part" hierarchy
  - Component class hierarchy
- **Role relations**—"has-a" relationships
  - Properties of classes

- Examples:
  - The concept of a book
  - The concept of a university
  - The concept of a wine

- Knowledge base of ontology elements and specific **instances**

Reasoning with ontologies

- "isa" and "ispart" relationships are **transitive**
  - If *X* isa *Y* and *Y* isa *Z* then *X* isa *Z*
  - If *X* ispart *Y* and *Y* ispart *Z* then *X* ispart *Z*
- **Inheritance** of properties
  - If *Z* hasa (part or role) *Y* and *X* isa *Z* then *X* hasa *Y*

- Examples from the real world
  - Human isa primate, primate isa mammal
  - Primate haspart arm, arm haspart hand
  - Crime hasa victim, murder is a crime

Ontology design

- Ontologies define **entities**, **events**, and **relations** as top-level categories
- Every ontology includes **taxonomy**
  - Tangled tree of ISA-linked concepts
- True ontology defines **structure** of each concept or class
  - **Slots**—properties, parts, and roles (whose fillers are also concepts)
- Some include **logical rules** of conceptual syntax
  - How to form more complex concepts from more primitive ones using various operators
  - E.g., restrictive modification such as middle-aged adult

Types of concepts and classes

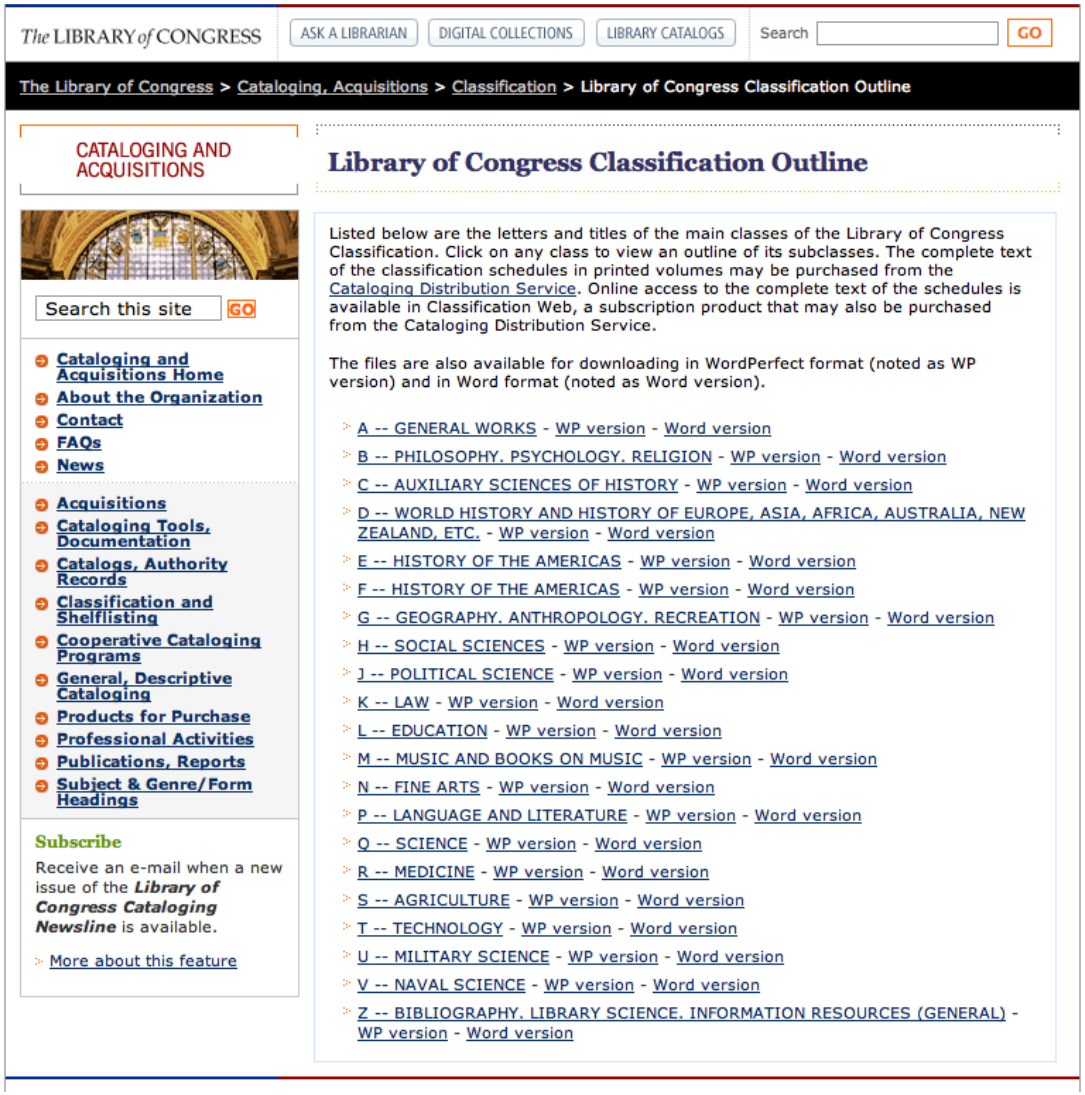
- Entity classes
  - person, number, chair
    chair has a seat part
    person has a gender
- Event classes
  - election, concert, murder
    events have a time property
    physical events have location and time
- Relation classes
  - employment, marriage, above
    (traditional) marriage has roles: husband, wife
    filler class for husband = male human
    filler class for wife = female human
- Goal of **concept-oriented ontology design**—formally represent *meaning* so a computer can manipulate it

The importance of taxonomies

- Topic taxonomy (Library Science)
  - Supports **search**
- Taxonomy of living things (Biology)
  - **Expresses and communicates** scientific knowledge
- Concept-oriented taxonomy (AI)
  - Supports **reasoning**
- Domain taxonomy (Information Systems; semantic web)
  - Supports **intelligent IR** and application design

- LCC—Library of Congress Classification outline (printed in 41 volumes)
  - Taxonomy for library classification based on topic

LCC class hierarchy



LCC subclasses…

LIBRARY OF CONGRESS CLASSIFICATION OUTLINE

CLASS H - SOCIAL SCIENCES

(Click each subclass for details)

Subclass H	Social sciences (General)
Subclass HA	Statistics
Subclass HB	Economic theory. Demography
Subclass HC	Economic history and conditions
Subclass HD	Industries. Land use. Labor
Subclass HE	Transportation and communications
Subclass HF	Commerce
Subclass HG	Finance
Subclass HJ	Public finance
Subclass HM	Sociology (General)
Subclass HN	Social history and conditions. Social problems. Social reform
Subclass HQ	The family. Marriage. Women
Subclass HS	Societies: secret, benevolent, etc.
Subclass HT	Communities. Classes. Races
Subclass HV	Social pathology. Social and public welfare. Criminology
Subclass HX	Socialism. Communism. Anarchism

…and sub-subclasses

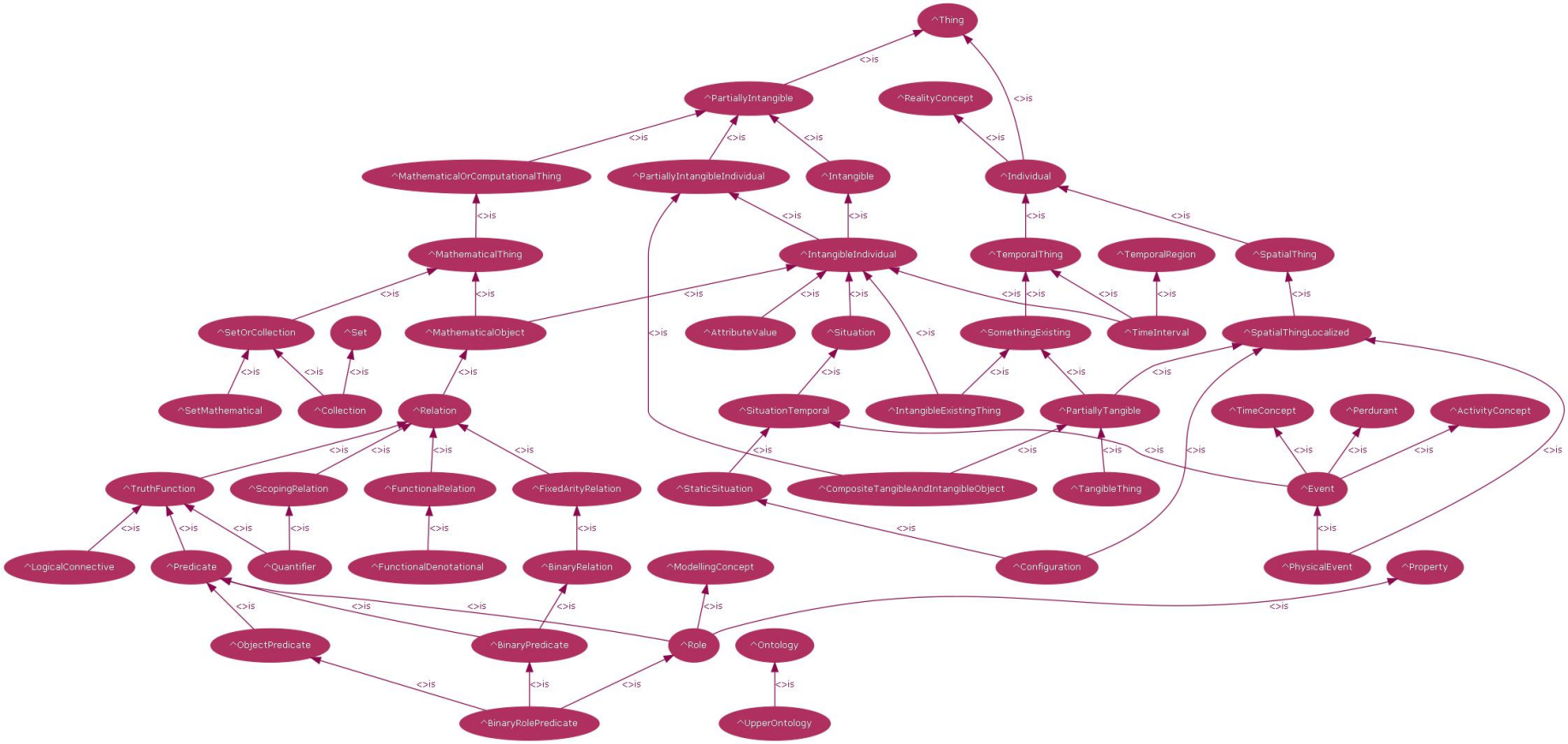
Subclass HB

HB1-3840	Economic theory. Demography
HB71-74	Economics as a science. Relation to other subjects
HB75-130	History of economics. History of economic theory Including special economic schools
HB131-147	Methodology
HB135-147	Mathematical economics. Quantitative methods Including econometrics, input-output analysis, game theory
HB201-206	Value. Utility
HB221-236	Price
HB238-251	Competition. Production. Wealth
HB501	Capital. Capitalism
HB522-715	Income. Factor shares
HB535-551	Interest
HB601	Profit
HB615-715	Entrepreneurship. Risk and uncertainty. Property
HB801-843	Consumption. Demand
HB846-846.8	Welfare theory
HB848-3697	Demography. Population. Vital events
HB3711-3840	Business cycles. Economic fluctuations

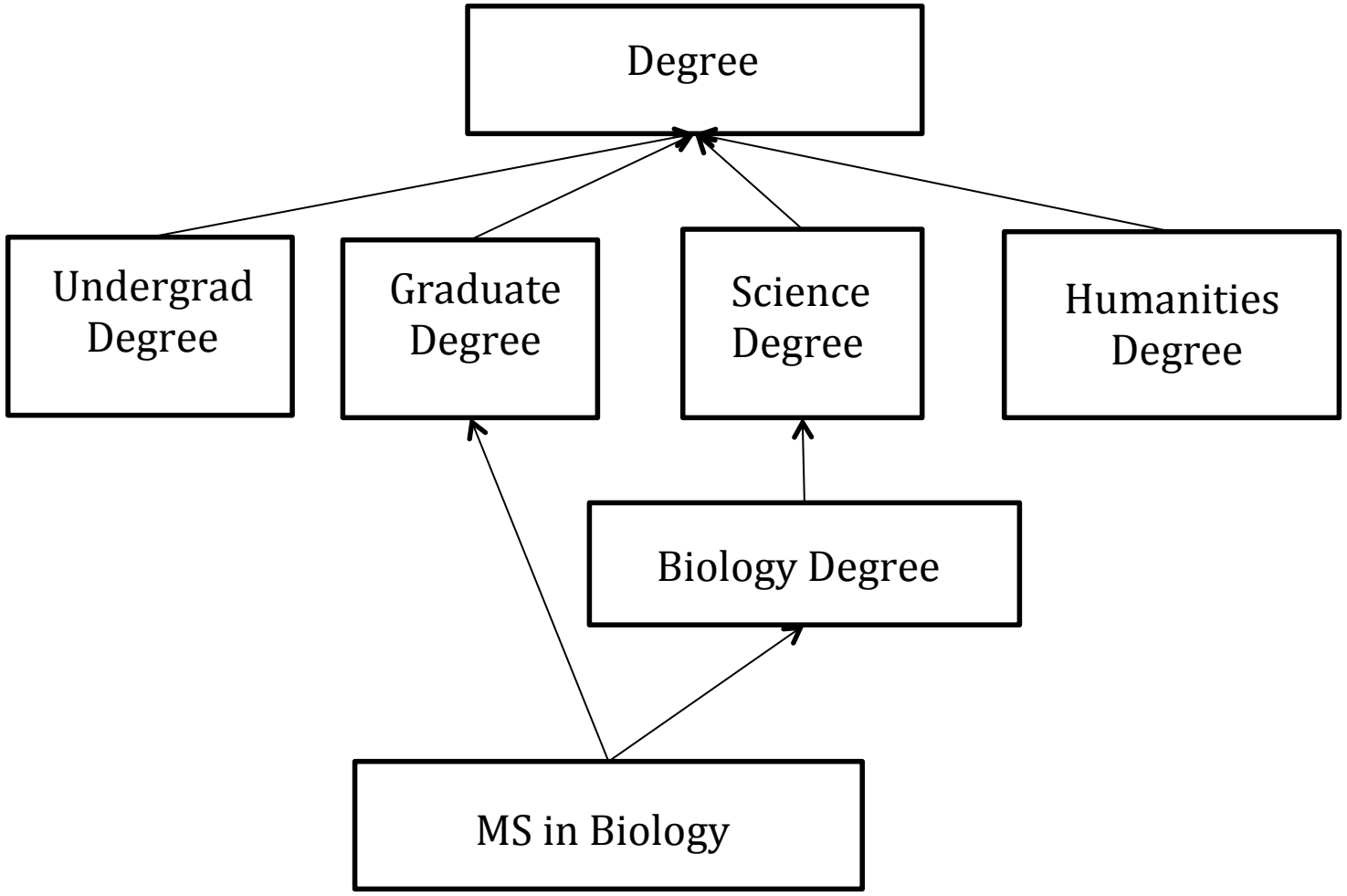
General v. domain specific ontologies

- **General knowledge** ontology—need frameworks for formalizing
  - Physical attributes—substances (liquid, solid, gas), shapes, etc.
  - Time—points, intervals, units, durations, time properties of events, axioms for temporal reasoning
  - Places and positions; spaces and objects
  - Qualities, quantities, and measurements
  - Motion, change, and causality
  - Human activities, motivations, typical behavior
- Requires hundreds of thousands/millions of concepts and associations
- For specific tasks create **domain** and **task** ontologies—manageable number of concepts

Cyc upper ontology



University degree taxonomy



Ontologies are everywhere!

- Every database schema is a domain ontology
- Every time you design an OO program with some classes, you are creating a domain or task ontology
- Many websites embody (implicitly or explicitly) a domain ontology
  - University web site has a concept—degree programs—arranged in a (possibly tangled) taxonomy

Knowledge engineering methodology

- Seven steps to creating an ontology
  1. Determine **domain** and **scope**
  2. Consider **reusing** existing ontologies
  3. **Enumerate** important concepts
  4. Define the classes and class hierarchy (**taxonomy**)
  5. Define the properties, parts, and roles (**slots**)
  6. Define the **facets/constraints** of the slots
  7. Create instances

- Iterative process—revise and refine evolving ontology
- Example—wine and food ontology for pairing wine with meals
  - Applications—auto sommelier in restaurants, matching wine cellar inventory to upcoming menus or cookbooks

Determine domain and scope

- What is the domain that the ontology will cover?
- For what are we going to use the ontology?
- For what types of questions should the information in the ontology provide answers (competency questions)?

Competency questions

- Questions that a KB based on the ontology should be able to answer
  - Litmus test for later **evaluation**
- Food and wine application
  - Which wine characteristics should I consider when choosing a wine?
  - Is Bordeaux a red or white wine?
  - Does Cabernet Sauvignon go well with seafood?
  - What is the best choice of wine for grilled meat?
  - Which characteristics of a wine affect its appropriateness for a dish?
  - Does a flavor or body of a specific wine change with vintage year?
  - What were good vintages for Napa Zinfandel?

Consider reusing existing ontologies

- Why reuse other ontologies?
  - Save effort
  - Interact with tools that use other ontologies
  - Use ontologies that have been validated through use in applications

What to reuse?

- Upper ontologies (general knowledge)
  - IEEE Standard Upper Ontology (http://suo.ieee.org)
  - Cyc (www.cyc.com)

- General ontologies
  - DMOZ (www.dmoz.org)
  - WordNet (http://wordnet.princeton.edu/)
- Domain specific ontologies
  - UMLS semantic Net
  - GO (Gene Ontology) (www.geneontology.org)

Enumerate important terms

- What are the concepts we need to talk about?
- What are the properties of these concepts?
- What do we want to say about the concepts?

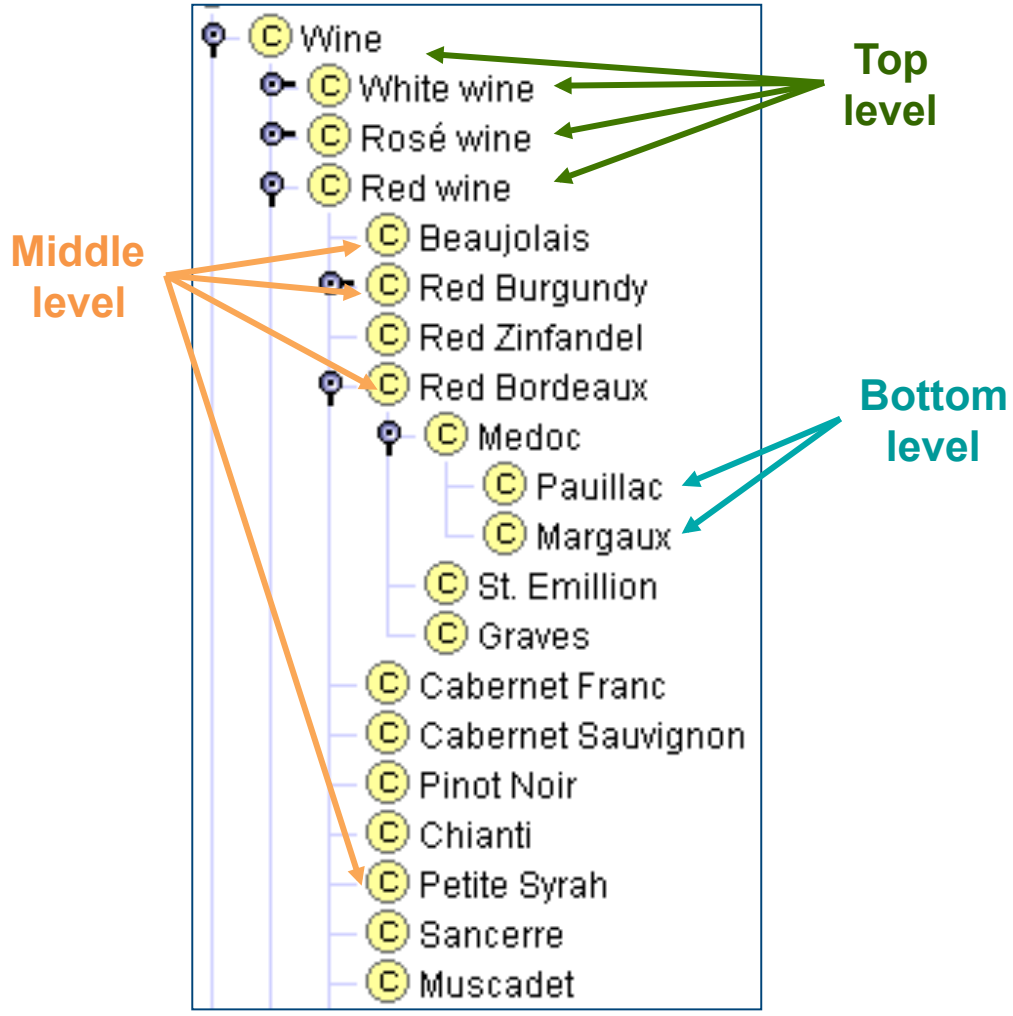
Enumerating terms: wine ontology example

- Wine, grape, winery location
- Wine color, wine body, wine flavor, sugar content
- White wine, red wine, Bordeaux wine
- Food, seafood, fish, meat, vegetables, cheese

Define classes and the class hierarchy

- A class is a concept in the domain
  - Class/subclass structure of wines
  - Class/subclass structure of wineries
- Class is a collection of elements with similar properties
- Instances of classes
  - A particular glass of California wine you'll have for lunch

Levels in the wine hierarchy



Modes of development

- **Top-down**
  - Define the most general concepts first and then specialize them
- **Bottom-up**
  - Define the most specific concepts first and then organize them in more general cases
- **Combination**
  - Define the more salient concepts first and then generalize and specialize them

Document everything!

- Classes (and slots) usually have documentation
  - Describing the class in natural language
  - Listing domain assumptions relevant to the class definitions
  - Listing synonyms
- Documenting classes and slots is as important as documenting and commenting computer programs!

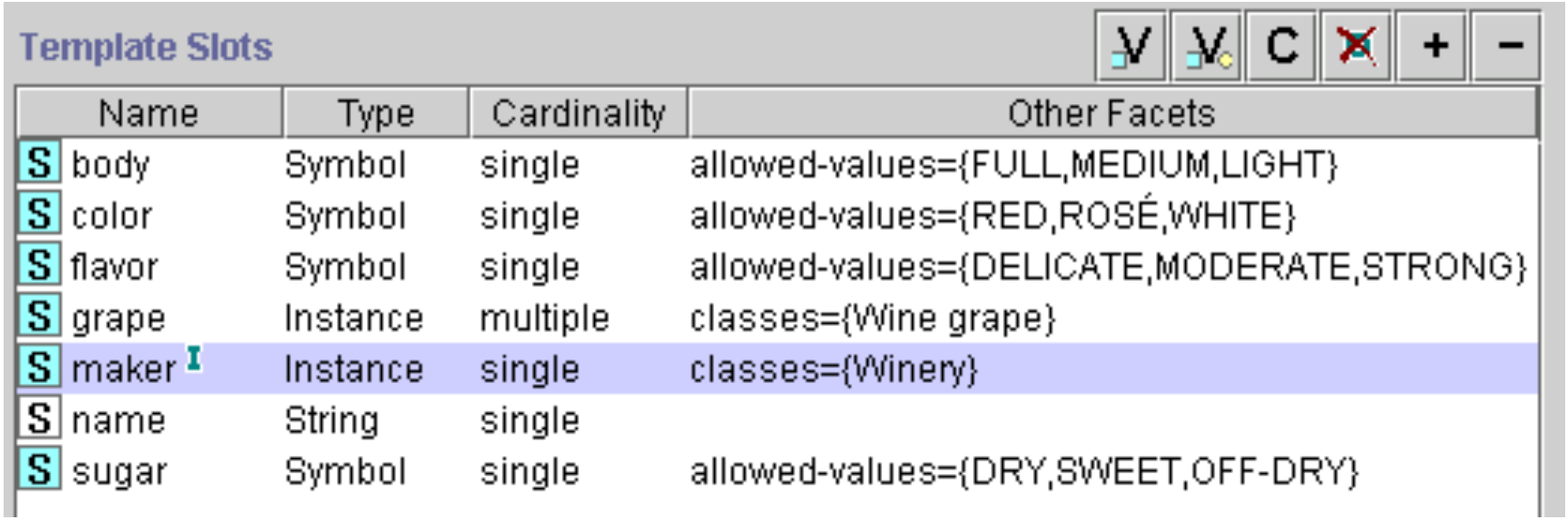
Define properties of classes (i.e., slots)

- Slots describe attributes of instances of a class and relations to other instances
- Example
  - Each wine will have color, sugar content, producer, etc.

Slots: Properties, parts, and roles

- Types of properties
  - Intrinsic (i.e., flavor and color of wine)
  - Extrinsic (i.e., name and price of wine)
  - Roles/relationships (producer of the wine)
- Simple and complex properties
  - Simple properties (attributes)—contain primitive values, i.e., strings and numbers
  - Complex properties—contain or point to other **concepts**, e.g., a winery instance

Slots for the wine class



Template Slots

Name	Type	Cardinality	Other Facets
S body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
S color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
S flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
S grape	Instance	multiple	classes={Wine grape}
S maker	Instance	single	classes={Winery}
S name	String	single	
S sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

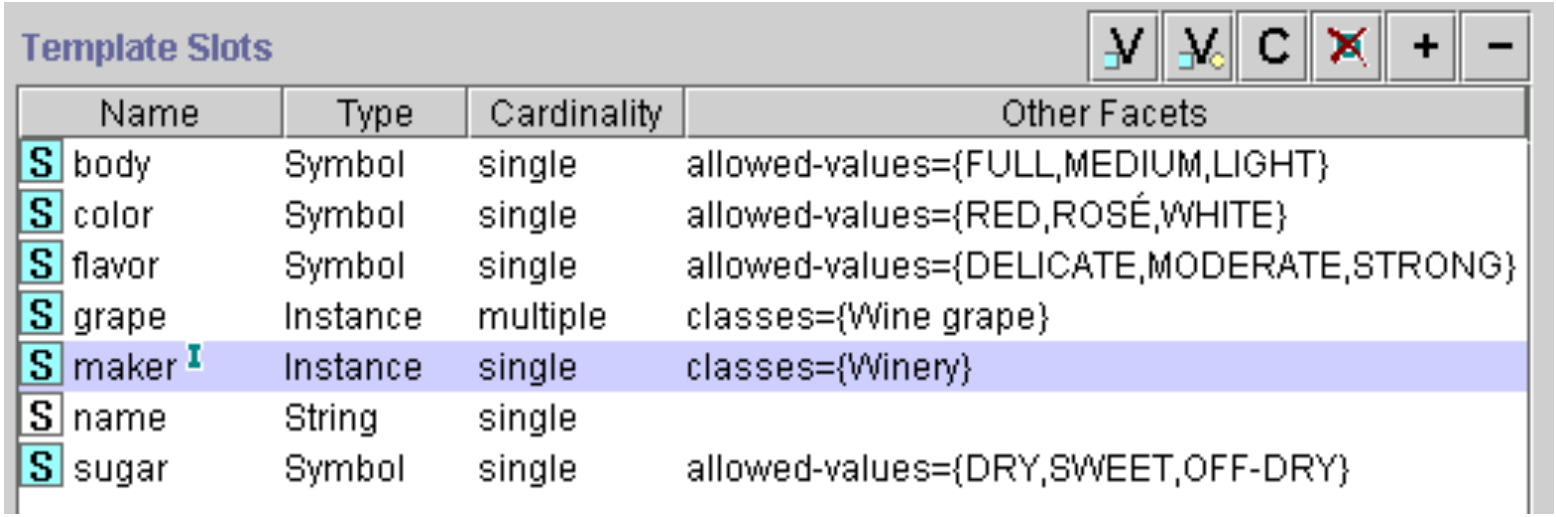
Slots and class inheritance

- Subclass **inherits** all slots from the superclass
  - If a wine has a name and flavor, a red wine also has a name and flavor
- If a class has **multiple** superclasses, it inherits from all of them
  - Port is both a dessert wine and a red wine—inherits "sugar content: high" from the former and "color: red" from the latter

Define constraints of the slots

- Property constraints (**facets**) describe or limit the set of possible values for a slot
  - The **name slot** of a wine has a type facet of string
  - The **producer slot** has a type facet instance Winery
  - The **location slot** of a winery has a cardinality facet of 1

Facets for slots in the wine class



Template Slots

Name	Type	Cardinality	Other Facets
body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
grape	Instance	multiple	classes={Wine grape}
maker	Instance	single	classes={Winery}
name	String	single	
sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

Common facets

- Slot **cardinality**—the number of values a slot has
- Slot value **type**—the data type of values a slot can take, e.g., string, number, instance of <class>, etc.
- **Minimum** and **maximum** value—range of values solt whose data type is numeric
- **Default** value—the value a slot has for an instance unless a value is explicitly specified

Facet value types

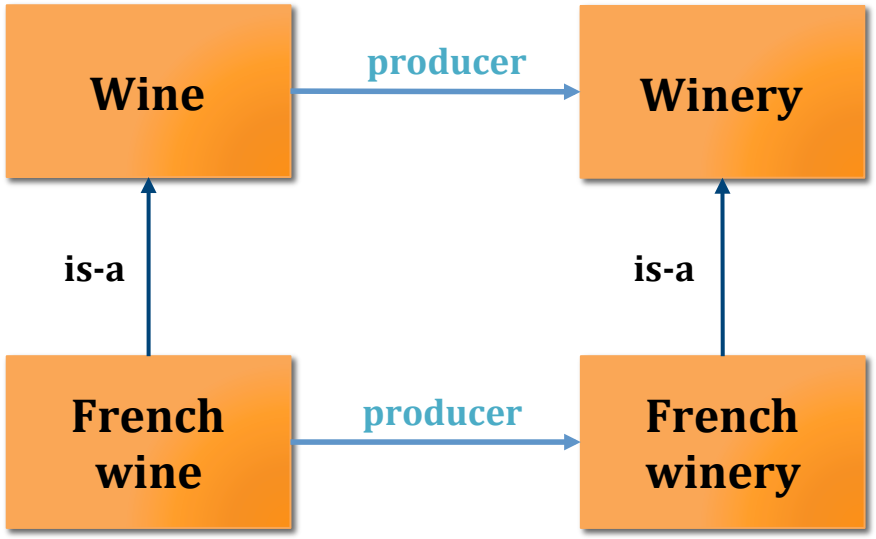
- **String**: string of characters (“Chateau Lafite”)
- **Number**: an integer or a float (15, 4.5)
- **Boolean** : a true/false flag
- **Enumerated** type: a list of specific allowed values (high, medium, low)
- **Complex** type: an instance of another class
  - Specify the class to which instances belong
  - The wine class is the value type for the slot “produces” at the winery class

Domain and range of a slot

- Domain—the class (or classes) that have the slot
  - More precisely: class (or classes) instances of which have the slot
- Range—the class (or classes) to which the slot values belong
  - i.e., the value type

Facets and class inheritance

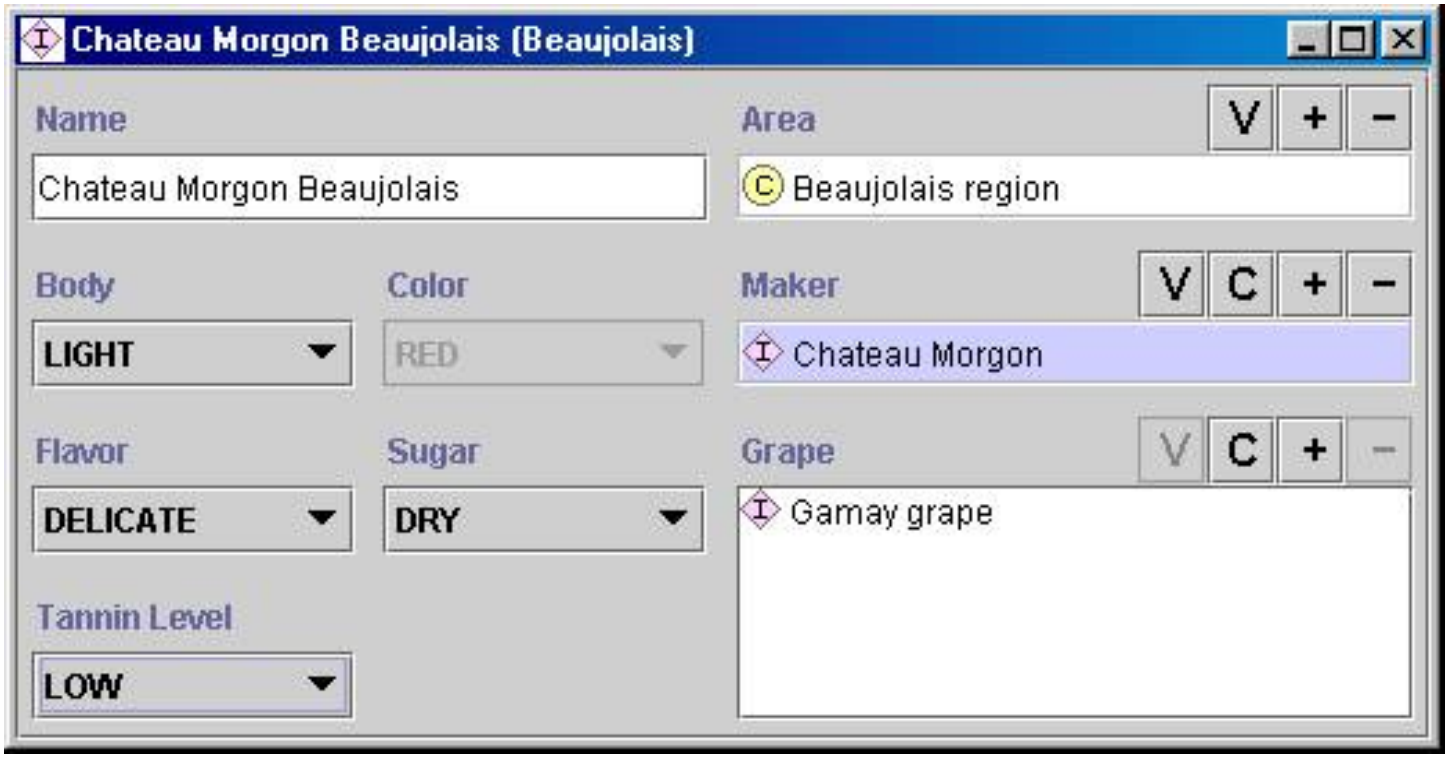
- Subclass inherits all the slots from the superclass
- Subclass can **override** facets to "narrow" the list of allowed values
  - Make cardinality range smaller
  - Replace a class in the range with a subclass



Create instances

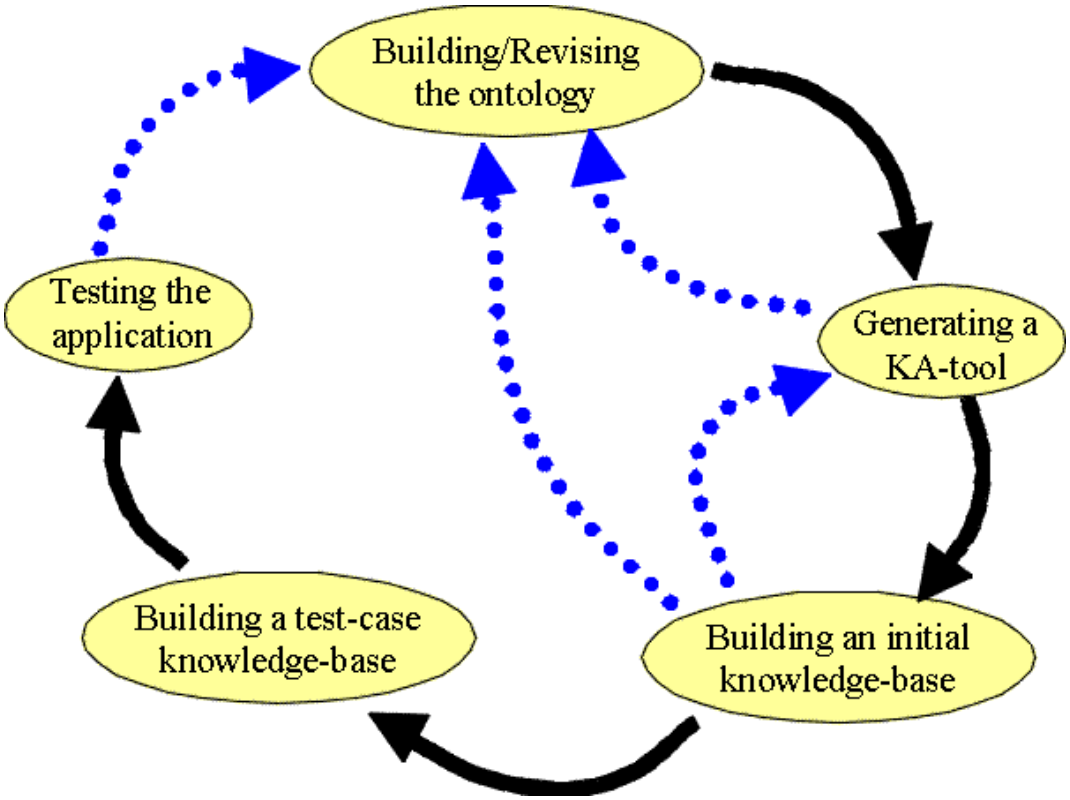
- Create an **instance** of a class
  - Class becomes a **direct type** of the instance
  - Any superclass of the direct type is a type of the object
- Assign slot values for the instance
  - Slot values must conform to facet constraints
  - **Knowledge-acquisition** tools often check that

Wine instance example



Ontology development is iterative

- Designing a good ontology requires **evaluation** and **revision** throughout the process



Reasoning in an ontology

- Axioms—formal **assertions** for inference in the ontology
  - Derive information about concepts, constraints on their structure, and their relations than are shown in the ontology structure
- **Constraints** on values of several properties
  - "The length of a brick is always greater than its height"
    $\forall x\ Brick(x) \Rightarrow Length(x) > Height(x)$
- Facts about the **relations**
  - "Every block is on something"
    $\forall x\ Block(x) \Rightarrow \exists y\ On(x,y)$
- Constraints on property or role values for **related objects**
  - "No block is on a smaller block"
    $\forall x,y\ [Block(x) \wedge Block(y) \wedge On(x,y) \Rightarrow Size(x) < Size(y)]$
- Represented in FOL

The semantic web

- Goal—to put information with **computer-processable meaning** (semantics) on the web
- E-commerce motivation and industry support
  - Would like to describe companies, products (entities) and transactions (events) to automate e-commerce (especially B2B e-commerce)
- Extends the web through use of
  - Standards
  - Markup Languages
  - Tools

Phases of the semantic web

- XML—eXtensible Markup Language
- RDF—Resource Description Framework
- OWL—Ontology Web Language
- Standards proposed by W3C: World Wide Web Consortium

Protégé ontology editor

- Free, open source ontology editor and knowledge engineering tool
- Ontology design is expensive!
  - Protégé designed to walk users through design and facilitate reuse and maintenance
- Two interfaces
  - **Protégé-Frames**—use this for your project
  - Protégé-OWL
- Currently used in clinical medicine and biomedical research!
- Download available from Blackboard or the Resources page on the website

Slot cardinality in Protégé

- Cardinality
  - Cardinality N means the slot **must have** N values
- Minimum cardinality
  - Minimum cardinality 1 means that the slot must have a value (**required**)
  - Minimum cardinality 0 means that the slot value is **optional**
- Maximum cardinality
  - Maximum cardinality 1 means that the slot can have **at most one** value (single-valued)
  - Maximum cardinality greater than 1 means the slot can have more than one value (multiple-valued slot)

Assignment 4—Ontology design

- **Objective:** Gain experience with knowledge engineering and write a research report
- **Project description:** Use the Protégé ontology editor (Protégé-Frames) to create a (partial) domain ontology for one of:
  - A town library
  - An art museum
  - A movie rental store
  - A campus bookstore (like ours)

  You should have several classes organized into a class hierarchy, with appropriate slots and facets. Use the wine and newspaper ontologies from the readings as a guide. **This assignment can be done in pairs.**

  Your written report should explain the major elements of your ontology and describe any problems you had or interesting issues you confronted in desigining the ontology. The report must identify the author(s), the course and have a title that includes the topic of your ontology.
- **Technical details:**
  - Naming your files—Both your ontology files and your report should have names that begin with all or part of one author's last name.
  - Protégé download —http://protege.stanford.edu/download/download.html
  - Protégé tutorial—http://protege.stanford.edu/doc/tutorial/get_started/admain.html