

Artificial Intelligence Programming

Ontologies

Chris Brooks

Department of Computer Science
University of San Francisco

16-2: Knowledge Engineering

- Logic provides one answer to the question of *how* to say things.
- It does not tell us *what* to say.
- Typically, this is the hard part.
- We want to give our agent enough knowledge to solve all of the problems we are interested in.
- The process of building a knowledge base is referred to as *knowledge engineering*
 - Many of the same principles as software engineering.

16-3: Knowledge Engineering

- The knowledge engineering process typically consists of the following steps.
 1. Determine the queries and types of facts that are available/allowed
 - For example, will the agent need to select actions or make conclusions, or just answer questions from a human user?
 - Will we ask existential queries, or just universal queries?
 2. Gather the relevant knowledge
 - Interview experts and find out how the domain works.
 3. Select a vocabulary of classes, functions, predicates and constants
 - This vocabulary is called an *ontology*
 4. Encode general knowledge about the domain
 - Formally represent the knowledge gathered in step 2.
 - This may require revisiting step 3.
 5. Encode specific queries or problems to be solved.
 - This may require returning to steps 1, 2, 3, or 4.

16-4: Ontologies

- An ontology is a vocabulary that describes all of the objects of interest in the domain and the relations between them.
 - All of the relevant knowledge about a problem we are interested in.
- Ontologies allow knowledge about a domain to be shared between agents (including humans).
- This can allow knowledge to be reused more easily.
- Allows an agent to perform inference with its current knowledge.

16-5: Vocabulary

- An ontology consists of:
 - A set of *concepts* or *classes*
 - $Professor(Brooks), \forall x Professor(x) \rightarrow USFEmployee(x)$
- Instances of these classes.
- Relations between classes of objects, sometimes called *slots* or *properties* or *roles*.
 - $Salary(Brooks, \$500), Name(Brooks, "Chris")$
- Restrictions on slots (sometimes called *facets*)
 - $\forall x, y Professor(x) \wedge Salary(x, y) \rightarrow y < \1000

16-6: Ontologies vs OO design

- Classes are a primary focus of ontology design.
- In many ways, this looks like object-oriented design.
- We have classes and subclasses, and properties of classes that look like data members.
- However, slots have richer semantics than data members.
- A slot may attach to several classes at once.
- We can specify constraints on the values in a slot's range.
- Slots can exist without being assigned to a class.

16-7: Graphical Representations

- Logic is a very powerful representation language, but it can be difficult for humans to work with.
- In addition, the lack of structure between sentences in a KB can make inference difficult.
- A *semantic network* is a way to graphically represent and reason about some logical concepts.

16-8: Semantic Net example

- $SisterOf(Mary, John)$
- $Female(Mary)$
- $\forall x Female(x) \rightarrow Person(x)$
- $Male(John)$
- $\forall x Male(x) \rightarrow Person(x)$
- $\forall x Person(x) \rightarrow Mammal(x)$
- $\forall x Person(x) \rightarrow DefaultNumberOfLegs(x, 2)$
- $legs(John, 1)$

16-9: Inference in a Semantic Net

- Inference becomes easy in a semantic net; to answer questions about John, we follow the labeled edges emanating from the *John* node.
 - This is the same sort of inference done by modern OO languages to resolve inheritance.
- Strengths: Knowledge is easily visualized, relationships between objects are clearer.
- Weaknesses: only binary relations, no negation, disjunction, or existential quantifiers.
 - We can extend semantic nets to include these features, but we lose the transparency.
 - Instead, use a semantic net to model class/slot relations, and FOL (or something similar) to model rules.

16-10: Protege

- Protege is a Java-based graphical tool for constructing ontologies.
- Has a rich set of plugins for performing inference and visualizing data.
- Can export data in RDF and OWL for use with the Semantic Web.

16-11: Using Protege to build a simple ontology

- Let's make a simple ontology to describe the USF CS department.
- We begin by asking *competency questions* - these are questions we'd like our KB to be able to answer.
 - Who is taking CS662?
 - Which professors teach classes on Monday?
 - How many students are taking both CS662 and CS601?
 - What classes should one take before taking CS662?
 - Which professors assign the most work?

16-12: Classes

- As with OO design, we can work top-down, bottom-up, or in a combination of the two.
- Let's start by making a Person class.
- We can then subclass that to make Professors and Students.
- Students have GradStudent and UndergradStudent subclasses.
- Let's also make a separate class representing courses.

16-13: Slots

- Now we can start to fill in the slots.
- Let's start with Student
 - Students have a Name, an ID, and some courses they're taking.
 - Let's also add a "year" slot to Undergrads- this can be Freshman, Sophomore, Junior, or Senior
- Next, we can do Professors.
 - Professors have a name, a salary, and some courses they teach.
 - Wait! Maybe name should be in person instead of student and professor.
 - Then do course - course should have an "inverse slot" for taught by and enrolls

16-14: Instances

- Now we can begin to populate our knowledge base with instances of the classes we’ve created.
- To begin, add “Brooks” as a Professor with salary \$100,000 (I wish!)
- We can create a course object on the fly to represent CS662.
- Next, create a student (use your own name).

16-15: Forms

- We can now enter instances, but the default editor settings are not very helpful.
- In particular, seeing the symbol names for each instance is unhelpful.
- Forms will let us change that.
- Forms also let you customize or modify the UI that a domain expert will use to add instances.

16-16: Querying the KB

- The query pane lets us create, save and retrieve queries
- We can specify either AND or OR.
- Try finding all students enrolled in CS662.

16-17: Extending the KB

- A KB is only as useful as the data that it contains in it.
- Add instances for Wolber and Parr.
- Add courses CS601, CS690, and CS112.
- Add more students.
- Add a class called Sysadmins and populate it with Alex.
- Finally, add a class called RA - it should have USFEmployee and GradStudent as superclasses.