CS 8520: Artificial Intelligence

Knowledge Representation

Paula Matuszek
Spring, 2013
Introduction

• Knowledge Representation means:
  – Capturing human knowledge
  – In a form computer can reason about

• Why?
  – Model human cognition
  – Add power to search-based methods

• Actually a component of all software development
KR Introduction

• General problem in Computer Science
• Solutions = Data Structures
  – words, arrays
  – records
  – lists, queues
  – objects

• More specific problem in AI
• Solutions = knowledge structures
  – decision trees
  – logic and predicate calculus
  – rules: production systems
  – description logics, semantic nets, frames
  – scripts
  – ontologies
We’ve been here before!

• Informed search:
  – a heuristic for informed search is adding knowledge

• Constraint satisfaction
  – heuristics for choosing which constraint next

• Logical agents:
  – FOL is one of the oldest forms of knowledge representation in AI

• etc
Characteristics of a good KR:

• It should
  – Be able to represent the knowledge important to the problem
  – Reflect the structure of knowledge in the domain
    • Otherwise our development is a constant process of distorting things to make them fit.
  – Capture knowledge at the appropriate level of granularity
  – Support incremental, iterative development

• It should *not*
  – Be too difficult to reason about
  – Require that more knowledge be represented than is needed to solve the problem
Kinds of Knowledge

Things we need to talk about and reason about; what do we know?

• Objects
  – Descriptions
  – Classifications

• Events
  – Time sequence
  – Cause and effect

• Relationships
  – Among objects
  – Between objects and events

• Meta-knowledge
  Distinguish between knowledge and its representation
  • Mappings are not one-to-one
  • Never get it complete or exactly right
Representation Mappings

- Knowledge Level
- Symbol Level
- Mappings are not one-to-one
- Never get it complete or exactly right
Knowledge engineering!

• Modeling the “right” conditions and the “right” effects at the “right” level of abstraction is difficult
• Knowledge engineering (creating and maintaining knowledge bases for intelligent reasoning) is an entire field of investigation
• Research goal: automated knowledge acquisition and machine learning tools to fill the gap:
  – We would like intelligent systems which learn about the conditions and effects, just like we do!
  – We would like intelligent systems which learn when to pay attention to, or reason about, certain aspects of processes, depending on the context!
Kinds of KR

• decision trees
• logic and predicate calculus
• rules: production systems
• description logics, semantic nets, frames
• scripts
• ontologies
Decision Trees

• Knowledge captured as a series of questions and responses or decisions and outcomes
• Common in troubleshooting manuals, medical domains, etc.
• Well-known example: Animals
• Often a binary tree, but doesn’t need to be
Decision Trees

• Example: Guessing an animal

• Is your animal a mammal?
  – Yes: Is your animal a pet?
    • Yes: Your animal is a cat
    • No: Your animal is a lion.
  – No: Is your animal bigger than a breadbox?
    • Yes: Your animal is a bear
    • No: Your animal is a mouse.
Decision Trees

• Decision trees are relatively simple representations leading to a single conclusion or action
• Nodes represent questions/tests/decisions
• Arcs represent answers/results
• Often but not necessarily binary
• Familiar in troubleshooting, biological keys, etc.
• What are you having for dinner?
  – chicken
    • What are the sides?
      – Mashed potatoes
        » Do you like sweet?
        » yes
        » Here’s a nice pinot grigio
        » no
      – french fries
      – asparagus

  – beef

  – fish
Decision Trees: Advantages

• Easy to implement. We know a *lot* about trees in computer science.

• Explanations and the inference process are clearcut and easy to explain.

• Low startup cost.

• Capture simple domains well.

• Fast to do inference: it’s just a tree walker.

• Given cases with answers, can use machine learning to develop.
Decision Trees: Disadvantages

- Decision trees reflect a semi-procedural view of expertise which is often not a good match to a domain
  - Difficult to modify
  - Difficult to maintain “tree” shape, even if you allow multiple inheritance
- Intermediate state of a problem is only captured implicitly.
- Don’t scale well. For complex domain, difficult to elicit from expert, hard to maintain and debug.
- May give illusion of structure which doesn’t actually reflect the domain
Decision Tree for Choosing a Wine

• Suppose you are writing an app to be used in a wine store to help a customer choose a wine and you want to use a decision tree.
• What would it look like?
Decision Tree for Wine
Logic and Predicate Calculus

• We have already discussed this at length in conjunction with logical agents

• Very rich representation

• For big real-world problems has some significant issues:
  – very bushy inference
  – does not match human expert thinking very well
    • excluded middle
    • No good choice for “don’t know”
Production Rules

• Common formalism in expert systems

• Knowledge is represented as if-then rules:
  – if <condition>   (LHS, left hand side)
  – then <action>    (RHS, right hand side)

• If car won’t start,
  – then see if battery is dead.

• If a person is a student,
  – then a person has an ID card.
Rules Continued

• **LHS** may be a test, an observation, a symptom, an already-known fact.
  – If the printer won’t print
  – If power test is passed
  – If strep diagnosed

• **RHS** may be a new fact to be asserted, an action to take, a message
  – Then see if it has power
  – Then assert (power, yes)
  – Then give antibiotics
Inference with rules

• Production rules systems typically have three components:
  – the knowledge base or KB (the rules)
  – the fact base (details for this instance)
  – the inference engine (application which uses rules)

• Inference engine repeatedly applies rules from the KB to create additional facts until a stopping point is reached
Inference Engines

• As with logic, may be forward-chaining, backward-chaining. May also be mixed.
  – find rules which can be applied, add to agenda
  – pick a rule from the agenda and fire it, updating KB

• Conflict resolution is method of choosing rule from among those on agenda
  – recency, specificity, explicit priority
Some additional issues

• Non-monotonicity. A rule may retract a fact.
  – eg: If printer(unplugged), then plug it in and retract printer(unplugged
• Truth maintenance. Rules on the agenda may no longer “belong there”.
• Uncertainty
Rule-based System Exercises

• Create a small rule-based system for recommending a wine
• Create a small rule-based system for diagnosing why your car won’t start.
Rule-based Wine Expert

• Rules for wine?
• Rules for car?
• Which worked better for wine?
• Which worked better for car?
• Why?
Evaluation of Rule-based Inference
Evaluation of Rule-based Inference

• Advantages
  – Relatively fast
  – Captures natural human patterns
  – Modular
  – Can capture uncertainty and non-monotonicity
  – Restricted syntax simplifies editors, learning, etc.
Evaluation of Rule-based Inference

• Advantages
  – Relatively fast
  – Captures natural human patterns
  – Modular
  – Can capture uncertainty and non-monotonicity
  – Restricted syntax simplifies editors, learning, etc.

• Disadvantages
  – Neither sound nor complete
  – Requires conflict resolution
  – restricted syntax reduces expressiveness
  – System behavior reliant on conflict resolution strategy
  – adding new rules may produce unusual effects under conflict resolution
Structured Knowledge Representations

- Modeling-based representations reflect the structure of the domain, and then reason based on the model.
  - Semantic Nets
  - Frames
  - Scripts
- Sometimes called associative networks
Basics of Associative Networks

• All include
  – Concepts
  – Various kinds of links between concepts
    • “has-part” or aggregation
    • “is-a” or specialization
    • More specialized depending on domain

• Typically also include
  – Inheritance
  – Some kind of procedural attachment
Semantic Nets

- graphical representation for propositional information
- originally developed by M. R. Quillian as a model for human memory
- labeled, directed graph
- nodes represent objects, concepts, or situations
  - labels indicate the name
  - nodes can be instances (individual objects) or classes (generic nodes)
- links represent relationships
  - the relationships contain the structural information of the knowledge to be represented
  - the label indicates the type of the relationship
Nodes and Arcs

- Arcs define binary relationships that hold between objects denoted by the nodes.

\[ \text{mother(john,sue)} \]
\[ \text{age(john,5)} \]
\[ \text{wife(sue,max)} \]
\[ \text{age(max,34)} \]
\[ \ldots \]
Semantic Networks

• The ISA (is-a) relation is often used to link instances to classes, classes to superclasses.

• Some links (e.g. hasPart) are inherited along ISA paths.

• The semantics of a semantic net can be relatively informal or very formal
  – often defined at the implementation level.

Semantic Network Diagram:

- Animal
  - hasPart
  - isa
- Bird
  - isa
- Robin
  - isa
- Rusty
- Red
  - isa

ISA links:
- Animal to Bird
- Bird to Robin
- Robin to Rusty and Red

 ISA paths:
- Animal to Bird to Robin
- Animal to Bird to Rusty
- Animal to Bird to Red
Individuals and Classes

- Many semantic networks distinguish
  - nodes representing individuals and those representing classes
  - the “subclass” relation from the “instance-of” relation
Inference by Inheritance

• One of the main kinds of reasoning done in a semantic net is the inheritance of values along the subclass and instance links.

• Semantic networks differ in how they handle the case of inheriting multiple different values.
  – All possible values are inherited, or
  – Only the “lowest” value or values are inherited
Multiple inheritance

- A node can have any number of superclasses that contain it, enabling a node to inherit properties from multiple “parent” nodes and their ancestors in the network.

- These rules are often used to determine inheritance in such “tangled” networks where multiple inheritance is allowed:
  - If $X < A < B$ and both $A$ and $B$ have property $P$, then $X$ inherits $A$’s property.
  - If $X < A$ and $X < B$ but neither $A < B$ nor $B < A$, and $A$ and $B$ have property $P$ with different and inconsistent values, then $X$ does not inherit property $P$ at all.
Conflicting inherited values
Nixon Diamond

• This was the classic example circa 1980.
  – If Person is Nixon, is he a pacifist?

![Diagram showing relationships between Person, Quaker, Republican, and pacifist](image-url)
Description Logics

• Description logics provide a family of KR systems with a formal semantics.
  – E.g., KL-ONE, LOOM, Classic, …

• An additional kind of inference done by these systems is automatic classification
  – finding the right place in a hierarchy of objects for a new description

• Semantic Nets can be considered as a form of description logic.
Description Logics

• Notations to make it easier to describe definitions and properties of categories
• Taxonomic structure is organizing principle
• Subsumption: Determine if one category is a subset of another
• Classification: Determine the category in which an object belongs
• Consistency: Determine if membership criteria are logically satisfiable
Current Best-known Description Logic

• OWL
  – Ontology Web Language
  – A language for the semantic web
  – Flavors: OWL-Lite, OWL-DL, OWL full, OWL 2
  – W3C recommendation as of 11 December, 2012
  – http://www.w3.org/TR/owl2-overview/
Ontologies

• structuring knowledge in a useful fashion
• An ontology formally represents concepts in a domain and relationships between those concepts
• The concept originated in philosophy; a model of a theory of nature or existence.
• An ontology describes the things we want to talk about, including both objects and relationships
What Is An Ontology

• An ontology is an explicit description of a domain:
  – concepts
  – properties and attributes of concepts
  – constraints on properties and attributes
  – Individuals (often, but not always)

• An ontology defines
  – a common vocabulary
  – a shared understanding
Ontology Examples

• Taxonomies on the Web
  – Google Recipes, DMOZ

• Catalogs for on-line shopping
  – Amazon.com product catalog

• Domain-specific standard terminology
  – Unified Medical Language System (UMLS) and MeSH

• Broad general ontologies
  – Cyc
MeSH Tree Structures - 2008

1. Anatomy [A]
   - Animals [B01] +
   - Algae [B02] +
   - Bacteria [B03] +
   - Viruses [B04] +
   - Fungi [B05] +
   - Plants [B06] +
   - Archaea [B07] +
   - Mesomycetozoea [B08] +

2. Organisms [B]
   - Animals [B01] +
   - Algae [B02] +
   - Bacteria [B03] +
   - Viruses [B04] +
   - Fungi [B05] +
   - Plants [B06] +
   - Archaea [B07] +
   - Mesomycetozoea [B08] +

3. Diseases [C]
4. Chemicals and Drugs [D]
5. Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]
6. Psychiatry and Psychology [F]
7. Biological Sciences [G]
8. Natural Sciences [H]
9. Anthropology, Education, Sociology and Social Phenomena [I]
10. Technology, Industry, Agriculture [J]
11. Humanities [K]
12. Information Science [L]
13. Named Groups [M]
14. Health Care [N]
15. Publication Characteristics [V]
16. Geographicals [Z]
Why Develop an Ontology?

• To share common understanding of the structure of information
  – among people
  – among software agents

• To enable reuse of domain knowledge
  – to avoid “re-inventing the wheel”
  – to introduce standards to allow interoperability
More Reasons

• To make domain assumptions explicit
  – easier to change domain assumptions (consider a genetics knowledge base)
  – easier to understand and update legacy data

• To separate domain knowledge from the operational knowledge
  – re-use domain and operational knowledge separately (e.g., configuration based on constraints)
Upper Ontologies

- Ultimate goal is to represent everything in the world!!
- Result is an upper ontology
Special- and General-purpose Ontologies

• Special-purpose ontology:
  – Designed to represent a specific domain of knowledge;
    • genetics (GO)
    • immune system (IMGT)
    • mathematics (Tom Gruber)

• General-purpose ontology:
  – Should be applicable in any special-purpose domain
  – Unifies different domains of knowledge

• Upper ontology provides highest level framework
  - all other concepts follow
Cyc Upper Ontology

• Cycorp released its core ontology for general use as OpenCyc.
• By far the largest general ontology created
• Cyc Upper Ontology satisfies two important criteria;
  – It is universal: Every concept can be linked to it
  – It is articulate: Distinctions are necessary and sufficient for most purposes
What is Cyc?

- A very large, manually entered *Knowledge Base*
  - ~3 million assertions (facts), inference rules, and definitions, in an ontology
  - To serve as a bias for automated learning
- An inference engine for answering arbitrary queries
- Various interfaces for gathering and sanity-checking yet more facts
  - Crowd-sourcing, NL interfaces, ...
- Automated knowledge collection
Cyc Ontology

Domain-Specific Knowledge
(e.g., Healthcare, Computer Security, Command and Control, Mortgage Banking, ...)

Domain-Specific Facts and Data

Slides courtesy of Cynthia Matuszek

Friday, February 22, 2013
Timeline

- 1990: “We ... anticipate a crossover from manual knowledge entry to automatic entry via natural language understanding later this decade.”

- 1994: “[Although] The impression one gets from its authors is that Cyc is well along the path ... As things stand right now there exists no objective measure of progress.”

- 2006: “Over the last twenty years, a sufficient core of common sense has been entered to allow it to begin increasing its own store of world knowledge. ... The original promise has not yet been fulfilled.”
What is it (actually) good for?

Currently, at various levels of quality:

- Integration of Heterogeneous Databases
- Knowledge-enhanced retrieval of captioned images
- Thesaurus Management
- Planning
- NLP
- Computer & Network Security
- Intelligent Tutoring
- Movie searches
- Information Retrieval

...etc.
The microLenat (µL)

- **Bogosity**: 1. [orig. CMU] The degree to which something is *bogus*. ... The agreed-upon unit of bogosity is the *microLenat*. Consensus is that ... one millionth of a Lenat is the largest unit practical for everyday use.
Bogosity: 1. [orig. CMU] The degree to which something is *bogus*. ... The agreed-upon unit of bogosity is the *microLenat*. Consensus is that **one millionth of a Lenat is the largest unit practical for everyday use**.

“We now place as high as 60% ... the chance that Cyc’s KB [will be] used by the next generation of AI programs, and its size and breadth will help make them more than theoretical exercises. No one doing research in symbolic AI in 1999 will want to be without a copy of Cyc.”
**Bogosity**: 1. [orig. CMU] The degree to which something is *bogus*. ... The agreed-upon unit of bogosity is the *microLenat*. Consensus is that ... **one millionth of a Lenat is the largest unit practical for everyday use**.

“We now place as high as 60% ... the chance that Cyc’s KB [will be] used by the next generation of AI programs, and its size and breadth will help make them more than theoretical exercises. No one doing research in symbolic AI in 1999 will want to be without a copy.

“Best: Cyc, or something similar, [will] serve as the foundation for the first true AI. ... No-one in the 21st century even considers buying a machine without common sense.”
**Bogosity:** 1. [orig. CMU] The degree to which something is *bogus*. ... The agreed-upon unit of bogosity is the *microLenat*. Consensus is that ...

*one millionth of a Lenat is the largest unit practical for everyday use.*

“We now place as high as 60% ... the chance that Cyc’s KB [will be] used by the next generation of AI programs, and its size and breadth will help make them more than theoretical exercises. No one doing research in symbolic AI in 1999 will want to be without a copy of Cyc.”

“Best: Cyc, or something similar, [will] serve as the foundation for the first true AI. … No one in the 21st century even considers buying a machine without common sense.”

“Doug talked about Cyc... and I had formed the impression from his enthusiastic description that Cyc would be able to answer a reasonable percentage of questions at this level of knowledge at least... [and] could be tested in English.”
A “Decade-Long Gamble”

- Obvious, gaping flaws:
  - Limited incorporation of statistical methods
  - Huge, opaque, fundamentally noisy system
  - **Very** limited demonstration of ML usefulness
  - Disdained / overhyped

- But also:
  - Still producing papers
  - Still getting grants to do hard things (and delivering)
  - Still working towards “big AI”

- Still going – 30 years later
Developing an Ontology

• Consider Questions Like:
  – Which wine should I serve with seafood today?
  – What wines should I buy next Monday for my reception in Villanova, PA?
  – Is there a market for the products of another small winery in this area?
  – What online source is the best for wines for my party in Texas next fall?
What Do We Need to Know?

• what types of wines have been served at different occasions
  – events (occasions)
  – wines and winetypes
• grape types, wine locations and wineries.
• Soil properties, weather, climate
• ratings, preferences.
• retailers
• foods
Which wine should I serve with seafood today?
Which wine should I serve with seafood today?
Which wine should I serve with seafood today?
Which wine should I serve with seafood today?
Which wine should I serve with seafood today?

A shared ONTOLOGY of wine and food

French wines and wine regions

California wines and wine regions
Which wine should I serve with seafood today?

A shared ONTOLOGY of wine and food

French wines and wine regions

California wines and wine regions

Friday, February 22, 2013
Wines and Wineries

Diagram:

- Winery
- Bordeaux
- Château Lafite Rothschild
- Pauillac
- Château Lafite Rothschild Pauillac

Connections:
- io
- best wineries
- maker
- produces
An Ontology Is Often Just the Beginning

Ontologies

Software agents

Problem-solving methods

Provide domain description

Declare structure

Databases

Knowledge bases

Domain-independent applications
What does an Ontology Look Like?

• Can be any knowledge representation

• Typically a description logic or associative network of some kind.
Ontology-Development Process

General approach:

- determine scope
- consider reuse
- enumerate terms
- define classes
- define properties
- define constraints
- create instances
Ontology-Development Process

General approach:

determine scope
consider reuse
enumerate terms
define classes
define properties
define constraints
create instances

Usually a highly iterative process. (Sound familiar?)
Preliminaries - Tools

• All screenshots in this presentation are from Protégé, which:
  – is a graphical ontology-development tool
  – supports a rich knowledge model
  – is open-source and freely available (http://protege.stanford.edu)

• Some other available tools:
  – Ontolingua and Chimaera
  – OntoEdit
  – OilEd
  – OpenCyc
Determine Domain and Scope

- 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 (competency questions)?
Competency Questions

• 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 Reuse

• Why reuse other ontologies?
  – to save the effort
  – to interact with the tools that use other ontologies
  – to use ontologies that have been validated through use in applications
What to Reuse?

• Ontology libraries
  – Protégé ontology library (protege.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary)
  – Biomedical ontologies (bioontology.org)
  – Many others: try Googling for your topic
  – http://www.dmoz.org/

• Upper ontologies
  – IEEE Standard Upper Ontology (suo.ieee.org)
  – Cyc (www.cyc.com)
Enumerate Important Terms

- What are the terms we need to talk about?
- What are the properties of these terms?
- What do we want to say about the terms?
Enumerating Terms: Wine

- wine, winery, food, grape, region, climate
Enumerating Terms - The Wine Ontology
Enumerating Terms - The Wine Ontology

wine, grape, winery, location,
Enumerating Terms - The Wine Ontology

wine, grape, winery, location,
wine color, wine body, wine flavor, sugar content
Enumerating Terms - The Wine Ontology

wine, grape, winery, location,
wine color, wine body, wine flavor, sugar content
white wine, red wine, Bordeaux wine
Enumerating Terms - The Wine Ontology

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
  - a class of wines
  - a class of wineries
  - a class of red wines
- A class is a collection of elements with similar properties
- Instances of classes
  - a glass of California wine you’ll have for lunch
Wine Classes

• red, white, rose
• sparkling
• French, German, Italian, ...
• type of grape
Wine Classes

• White, red, rose
  – Red:
    • Bordeaux, Chianti, Pinot Noir...
• French, California, Australian, Italian, ...
• Aperitif, dinner, dessert
Class Inheritance

- Classes usually constitute a taxonomic hierarchy (a subclass-superclass hierarchy)
- A class hierarchy is usually an IS-A hierarchy: *an instance of a subclass is an instance of a superclass*
- If you think of a class as a set of elements, a subclass is a subset
Class Inheritance - Example

• Apple is a subclass of Fruit
  
  *Every apple is a fruit*

• Red wines is a subclass of Wine

  *Every red wine is a wine*

• Chianti wine is a subclass of Red wine

  *Every Chianti wine is a red wine*
Levels in the Hierarchy
Levels in the Hierarchy

Bottom level
Levels in the Hierarchy
Levels in the Hierarchy

Top level

Middle level

Bottom level
Modes of Development

- **top-down** – define the most general concepts first and then specialize them
- **bottom-up** – define the most specific concepts and then organize them in more general classes
- **combination** – define the more salient concepts first and then generalize and specialize them
Documentation

• Classes (and properties) usually have documentation
  – Describing the class in natural language
  – Listing domain assumptions relevant to the class definition
  – Listing synonyms

• Documenting classes and properties is as important as documenting computer code!
Properties in a class definition describe attributes of instances of the class and relations to other instances.
Properties (Properties)

• Types of properties
  – “intrinsic” properties: inherent in the object
  – “extrinsic” properties: external, changeable
  – parts: components
  – relations to other objects:

• Simple and complex properties
  – simple properties (attributes): contain primitive values (strings, numbers)
  – complex properties: contain (or point to) other objects
Properties of Wines

• Properties?
  – “intrinsic” properties:
  – “extrinsic” properties:
  – parts:
  – relations to other objects:

• Simple and complex properties
  – simple properties (attributes):
  – complex properties:
Properties of Wines

• Each wine will have color, sugar content, producer, etc.

• “intrinsic” properties: flavor and color of wine

• “extrinsic” properties: name and price of wine

• parts: ingredients in a dish

• relations to other objects: producer of wine (winery)

• simple properties (attributes): color, flavor, etc

• complex properties: producer
Properties for the Class Wine

(in Protégé-2000)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Other Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>S body</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={FULL,MEDIUM,LIGHT}</td>
</tr>
<tr>
<td>S color</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={RED,ROSE,WHITE}</td>
</tr>
<tr>
<td>S flavor</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DELICATE,MODERATE,STRONG}</td>
</tr>
<tr>
<td>S grape</td>
<td>Instance</td>
<td>multiple</td>
<td>classes={Wine grape}</td>
</tr>
<tr>
<td>S maker</td>
<td>Instance</td>
<td>single</td>
<td>classes={Winery}</td>
</tr>
<tr>
<td>S name</td>
<td>String</td>
<td>single</td>
<td></td>
</tr>
<tr>
<td>S sugar</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DRY,SWEET,OFF-DRY}</td>
</tr>
</tbody>
</table>
Property and Class Inheritance

- A subclass inherits all the properties 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 properties from all of them

  *Port is both a dessert wine and a red wine. It inherits “sugar content: high” from the former and “color:red” from the latter*
Property Constraints

- Property constraints (restrictions) describe or limit the set of possible values for a property

  *The name of a wine is a string*
  *The wine producer is an instance of Winery*
  *A winery has exactly one location*
Restrictions for Properties at the Wine Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Other Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={FULL,MEDIUM,LIGHT}</td>
</tr>
<tr>
<td>color</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={RED,ROSE,WHITE}</td>
</tr>
<tr>
<td>flavor</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DELICATE,MODERATE,STRONG}</td>
</tr>
<tr>
<td>grape</td>
<td>Instance</td>
<td>multiple</td>
<td>classes={Wine grape}</td>
</tr>
<tr>
<td>maker</td>
<td>Instance</td>
<td>single</td>
<td>classes={Winery}</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>single</td>
<td></td>
</tr>
<tr>
<td>sugar</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DRY,SWEET,OFF-DRY}</td>
</tr>
</tbody>
</table>
Common Restrictions

- **Property cardinality** – the number of values a property has
- **Property value type** – the type of values a property has
- **Minimum and maximum value** – a range of values for a numeric property
- **Default value** – the value a property has unless explicitly specified otherwise
Some Restrictions for Wines

• Cardinality?
• Value?
• Default?
Common Restrictions: Property Cardinality

• Cardinality
  – Cardinality N means that the property must have N values

• Minimum cardinality
  – Minimum cardinality 1 means that the property must have a value (required)
  – Minimum cardinality 0 means that the property value is optional

• Maximum cardinality
  – Maximum cardinality 1 means that the property can have at most one value (single-valued property)
  – Maximum cardinality greater than 1 means that the property can have more than one value (multiple-valued property)
Common Restrictions: Value Type

- **String**: a string of characters ("Château Lafite")
- **Number**: an integer or a float (15, 4.5)
- **Boolean**: a true/false flag
- **Enumerated type**: a list of allowed values (high, medium, low)
- **Complex type**: an instance of another class
  - Specify the class to which the instances belong

*The Wine class is the value type for the property “produces” at the Winery class*
Restrictions and Class Inheritance

• A subclass inherits all the properties from the superclass

• A subclass can override the restrictions to “narrow” the list of allowed values
  – Make the cardinality range smaller
  – Replace a class in the range with a subclass
Create Instances

- Create an instance of a class
  - The class becomes a direct type of the instance
  - Any superclass of the direct type is a type of the instance

- Assign property values for the instance frame
  - Property values should conform to the constraints
  - Knowledge-acquisition tools often check that
An Instance Example

![Chateau Morgon Beaujolais](image)

**Name**
Chateau Morgon Beaujolais

**Area**
Beaujolais region

**Body**
LIGHT

**Color**
RED

**Flavor**
DELICATE

**Sugar**
DRY

**Maker**
Chateau Morgon

**Grape**
Gamay grape

**Tannin Level**
LOW
Defining Classes and a Class Hierarchy

• The things to remember:
  – There is no single correct class hierarchy
  – But there are some guidelines
  – The question to ask:
    • “Is each instance of the subclass an instance of its superclass?”

• Classes vs properties:
  – We could consider country as a class or a property
  – The question to ask:
    • Do I want to inherit other properties based on it?
Disjoint Classes

- Classes are disjoint if they cannot have common instances.
- Disjoint classes cannot have any common subclasses either.

**Red wine, White wine, Rosé wine are disjoint.**

**Dessert wine and Red wine are not disjoint.**
Classes and Their Names

• Classes represent concepts in the domain, not their names
• The class name can change, but it will still refer to the same concept
• Synonym names for the same concept are not different classes
  – Many systems allow listing synonyms as part of the class definition
Inverse Properties

Maker and Producer are inverse properties
Inverse Properties (II)

- Inverse properties contain redundant information, but
  - Allow acquisition of the information in either direction
  - Enable additional verification
  - Allow presentation of information in both directions

- The actual implementation differs from system to system
  - Are both values stored?
  - When are the inverse values filled in?
  - What happens if we change the link to an inverse property?
Limiting the Scope

• An ontology should not contain all the possible information about the domain
  – No need to specialize or generalize more than the application requires
  – No need to include all possible properties of a class
    • Only the most salient properties
    • Only the properties that the applications require
Limiting the Scope (II)

- Ontology of wine, food, and their pairings probably will not include:
  - Bottle size
  - Label color
  - My favorite food and wine
- An ontology of biological experiments will contain:
  - Biological organism
  - Experimenter
- Is the class Experimenter a subclass of Biological organism?
More Formal View of Ontologies

- Ontologies can be viewed, and reasoned about, from within the framework of First Order Logic.
- This is still controversial
  - exceptions
  - uncertainty
- But good formal approach
  - sound, complete reasoning
  - well-understood
- Textbook covers this in substantial detail
Categories

• Underlying concept of an ontology expressed in FOL is a category.

• Some important concepts for categories:
  – organization and inheritance
  – predicates and objects
  – partitions

• Some representation issues:
  – has-part or partOf
  – events, time, processes
  – physical and non-physical objects
Categories - Representation

- **Two choices for representation:**
  - **Predicate**
    - Basketball(b)
  - **Object**
    - Basketballs
    - Member(b, Basketballs)
    - Subset(Basketballs, Balls)
Categories - Organizing

• Inheritance:
  – All instances of the category Food are edible
  – Fruit is a subclass of Food
  – Apples is a subclass of Fruit
  – Therefore, Apples are edible

• The Class/Subclass relationships among Food, Fruit and Apples is a taxonomy
Categories - Partitioning

- **Disjoint**: The categories have no members in common
- **Exhaustive Decomposition**: Every member of the category is included in at least one of the subcategories
- **Partition**: Disjoint exhaustive decomposition
Categories - Partitioning

Disjoint({Animals, Vegetables})
Categories - Partitioning

Disjoint(\{\text{Animals, Vegetables}\})

Disjoint(s) \iff (\forall c_1,c_2 \ c_1 \in s \land c_2 \in s \land c_1 \neq c_2 \Rightarrow \text{Intersection}(c_1,c_2) = \{\})
Categories - Partitioning

Disjoint(\{Animals, Vegetables\})

Disjoint(s) \iff (\forall c_1, c_2 \; c_1 \in s \land c_2 \in s \land c_1 \neq c_2 \implies \text{Intersection}(c_1, c_2) = \{\})

ExhaustiveDecomposition

(\{Americans, Canadians, Mexicans\}, NorthAmericans)
Categories - Partitioning

Disjoint({Animals, Vegetables})

\[
\text{Disjoint}(s) \iff (\forall c_1, c_2 \ c_1 \in s \land c_2 \in s \land c_1 \neq c_2 \\
\Rightarrow \text{Intersection}(c_1, c_2) = \{\})
\]

ExhaustiveDecomposition

\[
(\{\text{Americans, Canadians, Mexicans}\}, \text{North Americans})
\]

\[
\text{ExhaustiveDecomposition}(s, c) \iff (\forall i \ i \in c \iff \exists c_2 \ c_2 \in s \land i \in c_2)
\]
Categories - Partitioning

Disjoint({Animals, Vegetables})
Disjoint(s) \iff (\forall c_1, c_2 \in s \land c_1 \neq c_2 \implies \text{Intersection}(c_1, c_2) = \{\})

ExhaustiveDecomposition
({Americans, Canadians, Mexicans}, NorthAmericans)

ExhaustiveDecomposition(s, c) \iff (\forall i \in c \iff \exists c_2 \in s \land i \in c_2)

Partition({Males, Females}, Animals)
Categories - Partitioning

Disjoint({Animals, Vegetables})

Disjoint(s) ⇔ (∀c1, c2 c1∈s ∧ c2∈s ∧ c1≠c2 ⇒ Intersection(c1, c2) = {})

ExhaustiveDecomposition

({{Americans, Canadians, Mexicans}, NorthAmericans})

ExhaustiveDecomposition(s, c) ⇔ (∀i i∈c ⇔ ∃c2 c2∈s ∧ i∈c2)

Partition({Males, Females}, Animals)

Partition(s, c) ⇔ Disjoint(s) ∧ ExhaustiveDecomposition(s, c)
Categories - More

• PartOf
  PartOf(Bucharest,Romania)
  PartOf(Romania,EasternEurope)
  PartOf(EasternEurope,Europe)
  PartOf(Europe,Earth)

• Composite Objects
  Biped(a) ⇒ ∃c1,c2,b  Leg(c1) ∧ Leg(c2) ∧ Body (b) ∧ PartOf(c1,a) ∧ PartOf(c2,a) ∧ PartOf(b,a) ∧ Attached(c1,b) ∧ Attached(c2,b) ∧ c1 ≠ c2 ∧ [∀c3 Leg(c3) ∧ PartOf(c3,a) ⇒ (c3=c1 ∨ c3=c2)]
Categories – And More

• Count Nouns and Mass Nouns
  – How many aardvarks? How many butters!?!?

• Intrinsic and Extrinsic Properties
  – Intrinsic properties belong to the very substance of the object; e.g. flavor, color, density, boiling point, etc.
  – Extrinsic properties change if the object is changed (cut in half); e.g. weight, length, shape, etc.
Generalized Events

• Combines aspects of space and time calculus

• Allows representation of events occurring in a space-time continuum

World War II is an event that happened in various geographic locations during a specific period of time within the 20th century.
Processes

• Discrete Events: the event is a whole and a part of the event is no longer the same event
• Processes can include subintervals; a part of a plane flight is still a member of the *Flying* class (aka *liquid events*)
• Stated more precisely: “Any subinterval of a process is also a member of the same process category.”
Intervals

- **Moment**: has temporal duration of zero
- **Extended Interval**: has temporal duration of greater than zero

\[
\text{Partition}\left(\{\text{Moments},\text{ExtendedIntervals}\},\text{Intervals}\right) \\
\text{Member}(i,\text{Moments}) \iff \text{Duration}(i) = \text{Seconds}(0).
\]
Intervals Ontology

Meet(i,j) ⇔ Time(End(i)) = Time(Start(j)).

Before(i,j) ⇔ Time(End(i)) < Time(Start(j)).

After(j,i) ⇔ Before(i,j).

During(i,j) ⇔ Time(Start(j)) ≤ Time(Start(i))
                  ∧ Time(End(i)) ≤ Time(End(j)).

Overlap(i,j) ⇔ ∃k During(k,i) ∧ During(k,j).
Mental Events and Mental Objects

- Knowledge about beliefs, specifically about those beliefs held by an agent
  - “Which agent knows about the geography of Maine?”
- Provides an agent the ability to reason about beliefs of agents
- However, need to define propositional attitudes, such as Believes, Knows and Wants as relations where the second argument is referentially opaque (no substitution of equal terms)
Reasoning Systems for Categories

• Categories are KR building blocks

• Two primary systems for reasoning:
  – Semantic Networks
    • Graphical aids for visualizing knowledge
    • Mechanisms for inferring properties of objects based on category membership
  – Description Logics
    • Formal language for constructing and combining category definitions
    • Algorithms for classifying objects and determining subsumption relationships
Reasoning with Default Information

• Open and Closed worlds
  – Open World: Information provided is not assumed to be complete, therefore inferences may result in sentences whose truth value is unknown
  – Closed World: Information provided is assumed complete, therefore ground sentences not asserted to be true are assumed false
  – Negation as Failure: A negative literal, not P, can be “proved” true if the proof of P fails
Knowledge Engineering

• Actually capturing the information from the human subject matter expert (SME) in any of these formats is difficult and time-consuming
  – An iterative process of add knowledge/test.
  – Often a knowledge engineer or ontological engineer works with the SME
  – “What is the system for?” is critical

• Automated learning of knowledge is a very active research field right now.

• KBs and KE vs statistics and ML