

Distributed Software Development

Ontologies and Information Interchange

Chris Brooks

Department of Computer Science
University of San Francisco

Outline

- Ontologies
 - What are they?
 - Why are they useful?
- OWL
- Standards for representing information
- Information Exchange
 - Old-school, top-down
 - New-school, bottom-up
- Ontologies and folksonomies

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.

Example

- Let's suppose that we're interested in selling books online.
- We would like to represent information about:
 - Books: title, author, ISBN, etc.
 - Authors: name, books written, URL, etc.
 - Customers: name, address, orders, payment info, etc.
 - Orders: books, date, address, shipper, etc.
- How might we represent this?
 - Note - this is a different question than "How would we store this?" the answer to that is most likely a relational DB.

Representing Knowledge

- More generally, how might we choose to represent knowledge so that a program can work with it?
- Keywords or tags. Quick and easy way to map terms onto data.
- Thesauri - Adds some descriptive power: more specific, more general, related.
- Taxonomy - Provides a classification structure - subclasses and superclasses.
- Ontology - provides potentially arbitrary relationships between data.

Ontologies again

- So an ontology is really just a formal way of describing the data in our system.
- Ontology-modeling languages usually include support for
 - Classes and instances
 - Inheritance
 - Relations between instances
 - Mechanisms for inference
- UML is an example of an ontology language.
- A big advantage of ontology-driven tools is the ability to reason about or to infer new facts about data.
- The challenge in a distributed environment is reconciling different ontologies or representations.

Goals

- What do we want to be able to do with a knowledge representation system?
 - Answer questions
 - Search
 - Discover patterns
 - Compare objects
 - Infer new facts/relations
 - Export knowledge to other applications
 - Easily annotate

Example: ebXML

- ebXML is an XML dialect created to provide a common language for business transactions.
- Meant to provide a common language for describing:
 - Business Processes
 - Core data components
 - Messaging
 - Registries and Repositories
- Example: Dealersphere
 - Built with ebXML, allows dealers to automate the process of credit checks, contract creation, searching for specific cars at other dealers, interoperating with each other.
 - Uses SOAP, J2EE, and ebXML

Representational Power

- What sort of representational power does XML provide us?

Representational Power

- What sort of representational power does XML provide us?
- Can assign metadata (or class information) to data
- Can specify sub-elements associated with elements
 - has-A relation
- Can specify actual data types
- Can specify sequences, sets
- Can uniquely identify objects

RDF

- RDF stands for *Resource Description Framework*
- Provides a way to describe properties that belong to objects.
- Much like a relational database.
- Can be serialized as XML, but it's easiest not to get hung up on the XML syntax initially.
 - Built to be created and consumed by machines.

RDF's data model

- Recall that XML produces a tree-structured data model.
- This can make it hard to represent some sorts of knowledge.
- How to denote that two elements share a sub-element?

Representation in XML

- For example:

```
<song>
  <artist>Doors</artist>
  <title>Break On Through</title>
  <album>Greatest Hits</album>
</song>
<song>
  <artist>Doors</artist>
  <title>Light My Fire</title>
  <album>Greatest Hits</album>
</song>
<song>
  <artist>Devo</artist>
  <title>Through Being Cool</title>
  <album>Greatest Hits</album>
```

RDF's data model

- RDF represents data in terms of *triples*
- Subject, property, value
- - “<http://www.cs.usfca.edu/brooks/S05classes/cs682/slides/rdf.ppt>”
title, “RDF Lab”
- Properties allow us to express relations between objects.
 - In AI, we called these things *predicates*

RDF's data model

- From a set of RDF triples, we can construct an RDF graph.
- Subject and value are nodes
 - Nodes can be
 - URIs - a generalized form of a URL
 - blank nodes - mostly useful as placeholders
 - literals - strings, values, etc.
 -
 - Properties are edges.

Resources

- An RDF document is a set of statements about *resources*
 - Documents, video clips, services
- A resource is something that has a location.
 - Referred to with a URI
- The subject of an RDF statement is a resource.

Literals

- The object of an RDF statement can be a resource or a literal.
- Literals are typically strings.
- For example:

```
<rdf:Description rdf:about='http://www.cs.usfca.edu/brooks'>  
  <dc:creator> Brooks </dc:creator>  
</rdf:description>
```

Properties and statements

- A property is a relation between a subject and an object.
- A statement is a subject, a property, and an object.
- This allows RDF statements to be placed in a graph model.
- We called this a semantic network in AI.

Reification

- It's also possible in RDF to make statements about statements.
- This process is called *reification*

```
<rdf:Description rdf:ID=''item10245''>  
  <externs:weight rdf:datatype=''&xsd;decimal''>2.4</externs:weight>  
</rdf:Description>
```

```
<rdf:Statement rdf:about=''#triple12345''>  
  <rdf:subject  
    rdf:resource=''http://www.example.com/2002/04/products#item10245''/>  
  <rdf:predicate rdf:resource=''http://www.example.com/terms/weight''/>  
  <rdf:object rdf:datatype=''&xsd;decimal''>2.4</rdf:object>  
  
  <dc:creator rdf:resource=''http://www.example.com/staffid/85740''/>  
</rdf:Statement>
```

- This lets you say things about who wrote a statement, when it was added, the validity, and so on.

Representational Power

- What additional representational power is provided by RDF?
- Arbitrary two-place relations between objects
- Inheritance of classes
- Inheritance of Properties
- Statements about statements
- Containers of objects
 - Sets, Bags, Sequences, Alternatives

Representational Power

- Believe it or not, this is still very limited.
- What other sorts of things might we want to represent?

Representational Power

- Believe it or not, this is still very limited.
- What other sorts of things might we want to represent?
- Class-specific property ranges. For example, in a music ontology, we might want to say that percussionists only play drums, while other performers can play any instrument.
- Disjointness. For example, male and female.
- Boolean combinations of classes. For example, we might want to define 'musician' to be the union of singer, performer, and composer.
- Cardinality restrictions. For example, a song has at least one composer, or a person has exactly one mother.
- Transitivity
- Inverse

OWL

- Web Ontology Language (OWL) provides a way to specify all of these concepts in what is called a *description logic*.
- Provides the ability to define concepts in terms of other concepts, along with restrictions and Boolean connectives.
- Tradeoffs: expressivity vs efficiency
 - OWL Full: most expressive, contains all the concepts described above. Inference is undecidable.
 - OWL DL: Restricts the ways in classes can be defined in terms of each other. Inference is complete, complexity unknown (NP-hard).
 - OWL Lite: disallows cardinality, enumerations, disjointness. Complexity: NP-complete, fast in practice.
- In practice, relatively efficient reasoners can be built for OWL DL.

OWL example

- (from Singh and Huhns)
- A trivial ontology defining animals
- Uses simple subclasses and properties
 - Disjointness goes beyond RDF
 - Object properties refine RDF properties; relate two objects

```
<owl:Class rdf:ID="Mammal">  
  <rdfs:subClassOf rdf:resource="#Animal"/>  
  <owl:disjointWith rdf:resource="#Reptile"/>  
</owl:Class>
```

```
<owl:ObjectProperty rdf:ID="hasParent">  
  <rdfs:domain rdf:resource="#Animal"/>  
  <rdfs:range rdf:resource="#Animal"/>  
</owl:ObjectProperty>
```

OWL inference

- What's attractive about OWL is the ability to perform inference.
- This allows us to discover class relationships that were not present in the data.
- Given the definition for the property hasParent and

```
<owl:Thing rdf:ID='Fido'>  
  <hasParent rdf:resource="#Rover"/>  
</owl:Thing>
```

- We can infer that Fido is an Animal

Using Inference to Build Class Relationships

- Knowledge engineering is a difficult process.
- People are usually not able to describe a complete ontology.
- Description Logic uses the following approach:
 - For a class, describe the necessary and sufficient conditions to be a member of that class.
 - Use automated inference to discover ontological structure.

Using Inference to Build Class Relationships

- Class construction happens in two stages:
- Explicitly (as in the examples above) or
- Anonymously, using
 - Restrictions
 - Set operators: `intersectionOf`, `unionOf`, `complementOf`, e.g.,

```
<owl:Class rdf:ID='SugaryBread'>  
  <owl:intersectionOf rdf:parseType='Collection'>  
    <owl:Class rdf:about='#Bread' />  
    <owl:Class rdf:about='#SweetFood' />  
  </owl:intersectionOf>  
</owl:Class>
```

Restrictions

- Restrictions govern the necessary and sufficient conditions for membership in a class.
 - someValuesFrom, allValuesFrom, hasValue, minCardinality, maxCardinality, etc.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasFather"/>
  <owl:maxCardinality rdf:datatype="xsd:nonNegativeInteger">
    1
  </owl:maxCardinality>
</owl:Restriction>
```

```
<owl:Restriction>
  <owl:onProperty rdf:resource='#bakes' />
  <owl:someValuesFrom rdf:resource='#Bread' />
</owl:Restriction>
```

- The reasoner uses restrictions to infer class membership and identity.

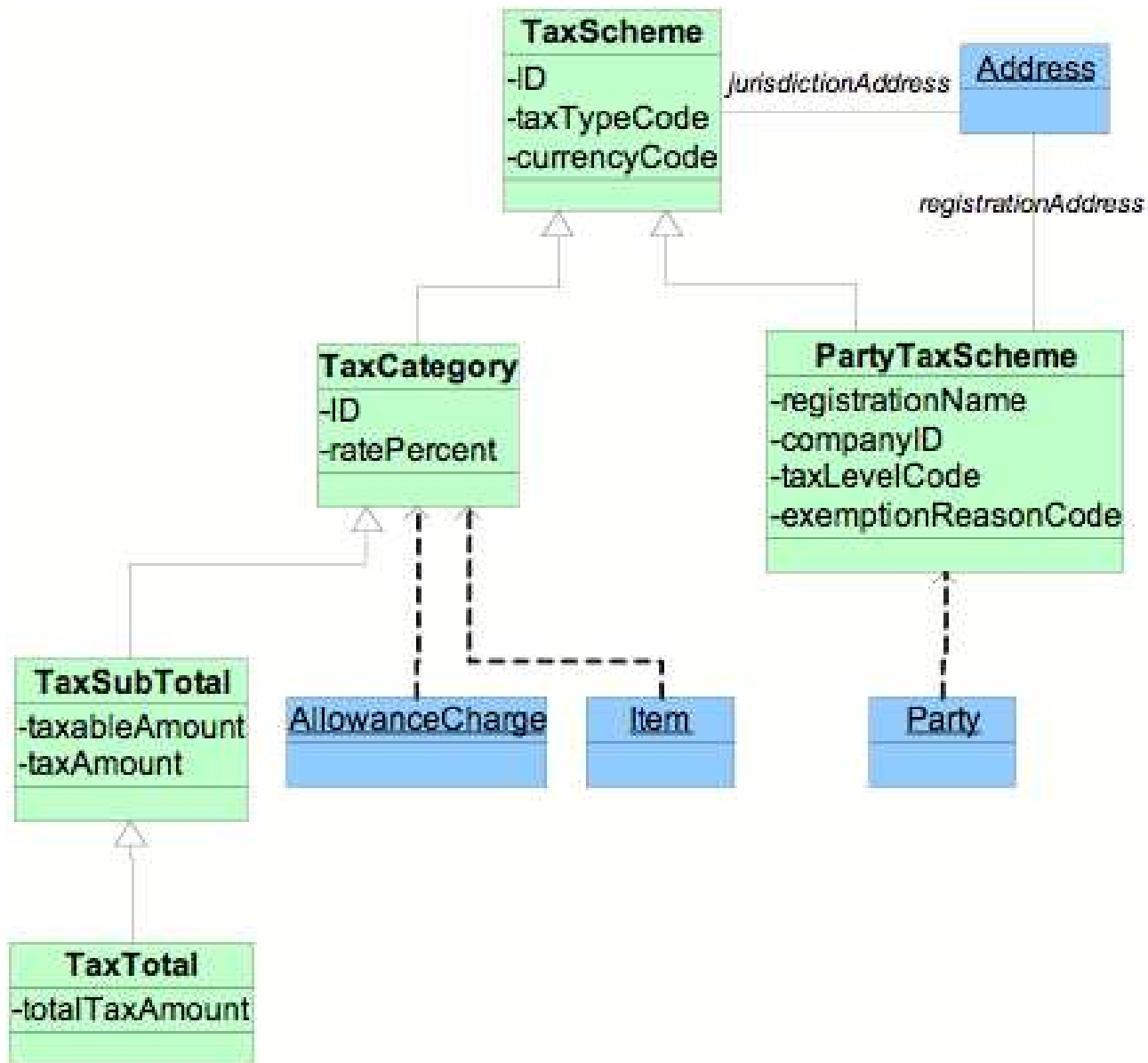
Information Interchange

- Ontologies can help in the description of services and the data they interchange.
- Problem: How do we get all parties involved to agree on the ontologies to use and to understand their proper use?
- Old approach: develop standards
- New approach: discover mappings

Developing Standards

- Several standard ontologies have been developed, at varying levels of generality.
- UBL - Universal Business Library. Meant to provide definitions for business logic.
- Cyc Upper Ontology. Meant to provide an ontology for describing “common-sense” knowledge.
- IEEE Standard Upper Ontology. Also meant to define general terms and concepts for specifying domain-specific ontologies.

UBL



Cyc

- The Cyc project has been going on for over 20 years.
- Goals:
 - Formally encode “common-sense” knowledge - the sorts of things everyone knows, but aren’t found in encyclopedias.
 - “If you spill water on yourself, you will be wet afterwards.”
 - Use this base to develop domain-specific ontologies and create agents that can reason deeply about a particular domain.
- Turns out to be a very hard problem
- Currently releasing part of their KB as open-source - 300,000 concepts, over 1 million facts.

Advantages of Standardization

- Once standards exist and are agreed upon,
 - Save time and improve performance
 - Enable the development of specialized tools.
 - Improve the longevity and reusability of a solution
 - Suggest directions for improvement

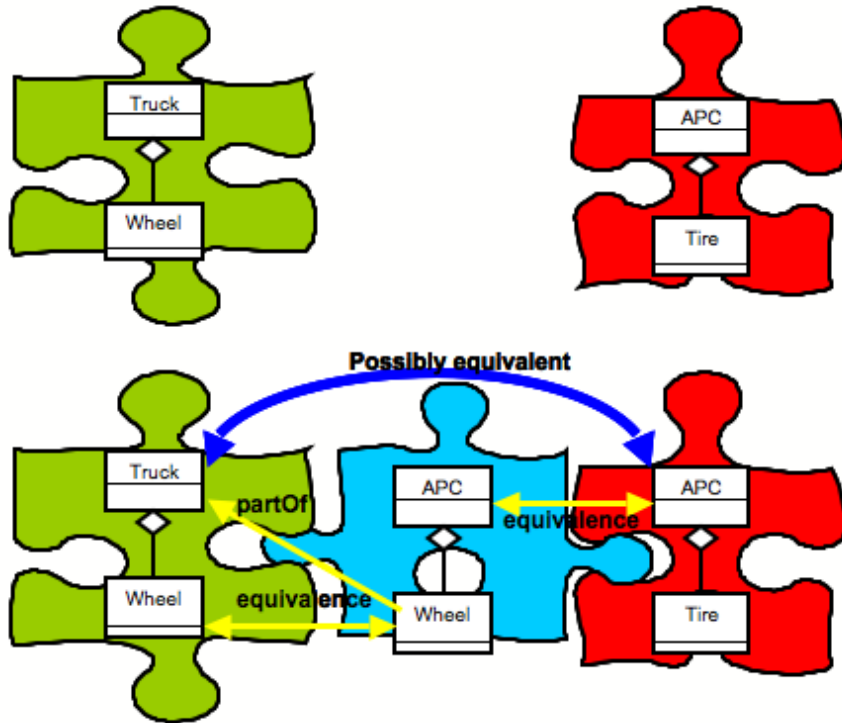
Problems with Standardization

- Standardization process is slow
- Still need to support legacy data
- Standards may not cover all necessary knowledge
- People will use their own ontologies for competitive advantage.
- “Heterogeneity is the normal state of affairs”

Mapping between ontologies

- Singh and Huhns describe an alternative approach to interoperability
- The construction of a *semantic bridge* between ontologies
- Example case:
 - You want your agent to get the results of a service for you
 - You express your needs in a query language
 - The service provider uses a different representation
 - Your agent must discover how to map your query into the provider's representation.

Example



- In the upper example, there's no clear way to map these two concepts.
- By using a third ontology, and information about how both of the other two ontologies map onto this ontology, we can infer a mapping between the original ontologies.

Merging Concepts in Ontologies

- The approach in Singh and Huhns consists of the following steps:
 - Bootstrap as much as possible with labeled data
 - Smart substring matching to identify points of similarity
 - Use graph structure to match other concepts.

Merging Concepts in Ontologies

- To match fragments of ontologies A and B, start by trying to find an ontology fragment C that shares some terminology with each of these.
- Ideally, there are many Cs, each describing some portion of a domain
 - Each C will have a different emphasis
 - Some concepts in common, some concepts different.
- This approach will work best for problems in which a number of individuals have constructed representations of a domain independently.

Merging Concepts in Ontologies

- Observation: when trying to determine how a concept in ontology A relates to a concept in ontology B, exactly one of seven relationships must hold:
 - subClass
 - superClass
 - partOf
 - hasPart
 - sibling
 - equivalence
 - other
- Problem - figure out which of these relationships is most likely.

Merging Concepts in Ontologies

- Begin with concepts known to match.
- Use string-matching to find concepts with identical/very similar names.
 - This provides a mapping at the lexical level
- At this point, there are some points of connection between each of a number of heterogenous ontologies.

Constructing a consensus ontology

- Next, Common structural information is inferred
- Concept a in ontology A is believed to map onto concept a' in ontology A'
- Concept b in ontology B is believed to map onto concept b' in ontology B'
- If a is a subclass of b in A , we can infer that a' is a subclass of b' in B'
- note: this is the sort of inference that's done in the Human Genome Project.

Constructing a consensus ontology

- Equivalence heuristics can be applied to concepts from each ontology that have common subclass and superclass relationships.
 - For example, in the experiment Singh and Huhns describe, they were able to discover that all Person concepts in ontology A fit the description of Human concepts in ontology B, and all Human concepts in ontology B fit the Person concepts from ontology A.
 - Therefore, Person and Human are equivalent concepts in the different ontologies.

Limitations

- This approach focuses on lexical and syntactic methods for determining similarity.
- More powerful approaches might use:
 - Other information sources (thesauri, dictionaries, etc)
 - More sophisticated schemes for resolving conflicts between ontologies
 - Interesting theoretical barriers to voting
 - Developing models of authority

“Correct” versus common usage

- An issue that comes up in the construction of consensus ontologies is the determination of meaning.
- Should a piece of information be categorized “correctly” or where most people would place it?
 - In other words, should an information provider place a piece of information at its “true” location, or
 - Where users are likely to search for it?
- e.g. If most users think that whales are fishes, should we place whale information in the fish category, or the mammal category?
- This is actually a manifestation of a much deeper linguistic problem - is meaning determined by authority, or usage?

Folksonomies

- Some researchers criticize the entire idea of ontologies
 - Real-world knowledge is messy and difficult to formalize
 - Meaning should be determined through consensus, rather than specified from the top down.
- Simple schemes for annotating data, when combined with large numbers of users, lead to a collaborative determination of the meaning associated with an annotation.
- This idea is often referred to as *folksonomy*.

Folksonomies

- The most well-known examples of systems that take this approach are shared-tagging systems such as del.icio.us and flickr.
- Users assign tags to photos or web pages.
- Argument: the meaning of a tag, such as “linux” is determined by the way in which it is used, rather than being specified by some external source.

Ontologies vs Folksonomies

- Are these ideas really completely different from each other?

Ontologies vs Folksonomies

- Are these ideas really completely different from each other?
- Most ontology research recognizes that a single standard ontology is not realistic.
- Both worlds are interested in achieving consensus through distributed creation and sharing of knowledge.
- Folksonomies typically eschew relational information
 - Makes systems easy to use, but limits expressivity
- Ontologies provide a great deal of relational power
 - More flexible, but more complex to use

Summary

- Ontologies are a formal way of representing the knowledge about a domain.
- OWL is a language for implementing ontologies specifically designed for the Web
 - Built on top of RDF
- Exchange of information between organizations is a fundamental problem in distributed systems
- Standards are great when they work, but can't be assumed
- Consensus through inference is an appealing alternative, but deep semantic matching is still a research topic.