Ontologies and Knowledge Sharing

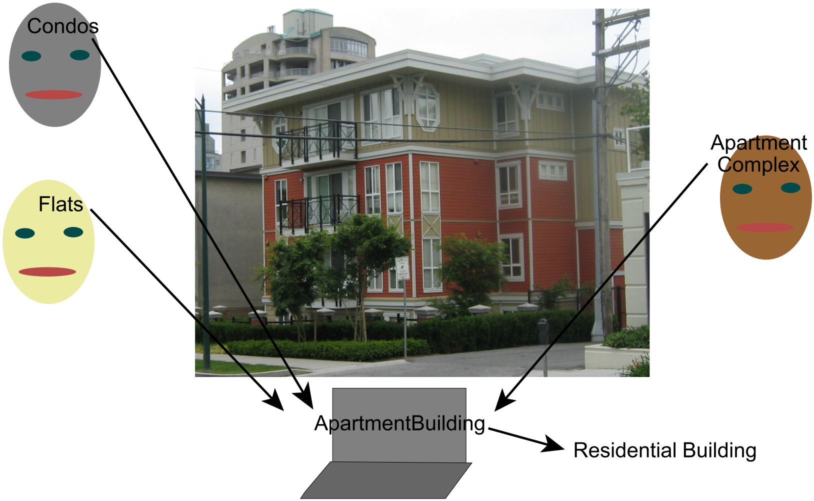
Building large knowledge respoitories:

- Knowledge often comes from multiple sources.
- Fields have their own terminology and division of the world.
- Systems evolve over time and it is difficult to anticipate all future distinctions that should be made.
- Designers must agree on what indiviuals, classes and relationships to represent. The world is not divided into individuals.
- It is often difficult to remember what notation means:
    - Given a symbol used in the computer, what does it mean?
    - Given a concept in someone's mind, what symbol to use?
        - Has the concept already been defined?
        - If already defined, what symbol has been used for it?
        - If not already defined, what can it be defined in terms of?

Knowledge Sharing

- A conceptualization is a map from the problem domain into the representation. A conceptualization specifies:
    - What sorts of individuals are being modeled
    - The vocabulary for specifying individuals, relations and properties
    - The meaning or intention of the vocabulary
- If more than one person is building a knowledge base, they must be able to share the conceptualization.
  $\longrightarrow$ challenge: inter-operability of separately designed knowledge bases.
- An ontology is a specification of a conceptualization.
  An ontology specifies the meanings of the symbols in an information system.

Mapping from a conceptualization to a symbol



Semantic Web

- Ontologies are published on the web in machine readable form.
- Builders of knowledge bases or web sites adhere to and refer to a published ontology:
  - A symbol defined by an ontology means the same thing across web sites that obey the ontology.
  - If someone wants to refer to something not defined, they publish an ontology defining the terminology.
    Others adopt the terminology by referring to the new ontology.
    In this way, ontologies evolve.
  - Separately developed ontologies can have mappings between them published.

Challenges of building ontologies

- They can be huge: finding the appropriate terminology for a concept may be difficult.
- How one divides the world can depend on the application. Different ontologies describe the world in different ways.
- People can fundamentally disagree about an appropriate structure.
- Different knowledge bases can use different ontologies.
- To allow KBs based on different ontologies to inter-operate, there must be mapping between ontologies.
- It has to be in user's interests to use an ontology.
- The computer doesn't understand the meaning of the symbols. The formalism can constrain the meaning, but can't define it.

Semantic Web Technologies Revisited

- RDF the Resource Description Framework is a language of triples, including the property `rdf:type` and containers (bags, lists, etc)
- RDF-S RDF Schema is RDF plus the class: `rdfs:Class`, and properties: `rdfs:domain`, `rdfs:range`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, ...
- Lots of alternative syntaxes: XML, Turtle, N-Triples, Json ...
- OWL the Web Ontology Language, defines some primitive properties that can be used to define terminology. (Uses multiple alternative syntaxes).

Main Components of an Ontology

- Individuals the things / objects in the world (not usually specified as part of the ontology)
- Classes sets of individuals
- Properties between individuals and their values

Individuals

- Individuals are things in the world that can be named. (Concrete, abstract, concepts, reified).
- Unique names assumption (UNA): different names refer to different individuals.
- OWL does *not* adopt the unique names assumption.
- The UNA is not an assumption you can universally make: "Lewis Carroll", "Charles Lutwidge Dodgson", "the author of Alice's Adventures in Wonderland" etc.
- Without the determining equality, we can't count! Joe's mother was in the room. Sam's cousin was there. Chris's football coach was there. How many people were in the room?
- Using OWL:

$$(i_1, \text{'owl:SameIndividual'}, i_2)$$

$$(i_1, \text{'owl:DifferentIndividuals'}, i_3)$$

Classes

- A class is a set of individuals. E.g., house, building, officeBuilding
- One class can be a subclass of another

  rdfs:SubClassOf(*house*, *building*)

  rdfs:SubClassOf(*officeBuilding*, *building*)

- The most general class is owl:Thing.
- Classes can be declared to be the same or to be disjoint:

  owl:EquivalentClasses(*house*, *singleFamilyDwelling*)

  owl:DisjointClasses(*house*, *officeBuilding*)

- Different classes are not necessarily disjoint.
  E.g., a building can be both a commercial building and a residential building.

Example Concepts in an Ontology

The following are some of the concepts in an ontology for documents.
`http://www.cs.umd.edu/projects/plus/DAML/onts/docmnt1.0.daml`

homepage	correspondence	publication
letter	periodical	article
book	email	magazine
journal	document	communication
workshopPaper	journalPaper	discussion
newspaper	PersonalHomepage	speech

Properties

- A property is between an individual and a value.
- A property has a domain and a range.

  rdfs:domain(*livesIn*, *person*)

  rdfs:range(*livesIn*, *placeOfResidence*)

- An *ObjectProperty* is a property whose range is an individual.
- A *DatatypeProperty* is one whose range isn't an individual, e.g., is a number or string.
- There can also be property hierarchies:

  rdfs:subPropertyOf(*livesIn*, *enclosure*)

  rdfs:subPropertyOf(*principalResidence*, *livesIn*)

Clicker Question

Suppose we are given the following triple as true:

```
years_eligibility 'rdfs:domain' student.
sam  years_eligibility  3).
```

Which is the following can we infer

- A Sam is a student
- B Sam could a student (but maybe isn't)
- C All students have value 3 for `years_eligibility`
- D We can infer nothing about whether Sam is a student

Clicker Question

Suppose we are given the following triples as true:

```
years_eligibility  'rdfs:domain'  student.
years_eligibility 'rdfs:domain' athlete.
sam years_eligibility 3.
```

Which is the following is true

A Sam is both a student and an athlete.

B Sam could be either student or an athlete.

C We can infer nothing about whether Sam is an athlete or a student

D There are no student athletes.

E The facts are inconsistent, and couldn't possible all be true

Properties (Cont.)

- One property can be inverse of another

  owl:InverseObjectProperties(*livesIn*, *hasResident*)

- Properties can be declared to be transitive, symmetric, functional, or inverse-functional.
  (Which of these are only applicable to object properties?)
- We can also state the minimum and maximal cardinality of a property.

  owl:minCardinality(*principalResidence*, 1)

  owl:maxCardinality(*principalResidence*, 1)

Property and Class Restrictions

- We can define complex descriptions of classes in terms of restrictions of other classes and properties.
  E.g., A homeowner is a person who owns a house.

$$homeOwner \subseteq person \cap \{x : \exists h \in house \text{ such that } x \; owns \; h\}$$

owl:subClassOf(homeOwner,person)

owl:subClassOf($homeOwner$,

    owl:ObjectSomeValuesFrom($owns$, $house$))

OWL Class Constructors

owl:Thing $\equiv$ all individuals
owl:Nothing $\equiv$ no individuals
owl:ObjectIntersectionOf$(C_1, \dots, C_k) \equiv C_1 \cap \cdots \cap C_k$
owl:ObjectUnionOf$(C_1, \dots, C_k) \equiv C_1 \cup \cdots \cup C_k$
owl:ObjectComplementOf$(C) \equiv \mathit{Thing} \setminus C$
owl:ObjectOneOf$(I_1, \dots, I_k) \equiv \{I_1, \dots, I_k\}$
owl:ObjectHasValue$(P, I) \equiv \{x : x\ P\ I\}$
owl:ObjectAllValuesFrom$(P, C) \equiv \{x : x\ P\ y \rightarrow y \in C\}$
owl:ObjectSomeValuesFrom$(P, C) \equiv$
$\{x : \exists y \in C \text{ such that } x\ P\ y\}$
owl:ObjectMinCardinality$(n, P, C) \equiv$
$\{x : \#\{y | xPy \text{ and } y \in C\} \geq n\}$
owl:ObjectMaxCardinality$(n, P, C) \equiv$
$\{x : \#\{y | xPy \text{ and } y \in C\} \leq n\}$

OWL Predicates

rdf:type$(I, C) \equiv I \in C$
rdfs:subClassOf$(C_1, C_2) \equiv C_1 \subseteq C_2$
owl:EquivalentClasses$(C_1, C_2) \equiv C_1 \equiv C_2$
owl:DisjointClasses$(C_1, C_2) \equiv C_1 \cap C_2 = \{\}$
rdfs:domain$(P, C) \equiv$ if xPy then $x \in C$
rdfs:range$(P, C) \equiv$ if xPy then $y \in C$
rdfs:subPropertyOf$(P_1, P_2) \equiv xP_1y$ implies xP_2y
owl:EquivalentObjectProperties$(P_1, P_2) \equiv xP_1y$ if and only if xP_2y
owl:DisjointObjectProperties$(P_1, P_2) \equiv xP_1y$ implies not xP_2y
owl:InverseObjectProperties$(P_1, P_2) \equiv xP_1y$ if and only if yP_2x
owl:SameIndividual$(I_1, \ldots, I_n) \equiv \forall j \forall k\ I_j = I_k$
owl:DifferentIndividuals$(I_1, \ldots, I_n) \equiv \forall j \forall k\ j \neq k$ implies $I_j \neq I_k$
owl:FunctionalObjectProperty$(P) \equiv$ if xPy_1 and xPy_2 then $y_1 = y_2$
owl:InverseFunctionalObjectProperty$(P) \equiv$
if x_1Py and x_2Py then $x_1 = x_2$
owl:TransitiveObjectProperty$(P) \equiv$ if xPy and yPz then xPz
owl:SymmetricObjectProperty $\equiv$ if xPy then yPx

Knowledge Sharing

- One ontology typically imports and builds on other ontologies.
- OWL provides facilities for version control.
- Tools for mapping one ontology to another allow inter-operation of different knowledge bases.
- The semantic web promises to allow two pieces of information to be combined if
  - they both adhere to an ontology
  - these are the same ontology or there is a mapping between them.

Example: Apartment Building

An apartment building is a residential building with more than two units and they are rented.

```
Declaration(ObjectProperty(:numberOfunits))
FunctionalObjectProperty(:numberOfunits)
ObjectPropertyDomain(:numberOfunits :ResidentialBuilding)
ObjectPropertyRange(:numberOfunits
                    ObjectOneOf(:two :one :moreThanTwo))

Declaration(Class(:ApartmentBuilding))
EquivalentClasses(:ApartmentBuilding
    ObjectIntersectionOf(
        :ResidentialBuilding
        ObjectHasValue(:numberOfunits :moreThanTwo)
        ObjectHasValue(:ownership :rental)))
```


Example: hotel ontology

Define the following:

- Room
- BathRoom
- StandardRoom - what is rented as a room in a hotel
- Suite
- RoomOnly
- Hotel
- HasForRent
- AllSuitesHotel
- NoSuitesHotel
- HasSuitesHotel

Top-Level Ontology

A top-level ontology

- provides a definition of *everything* at a very abstract level.
- provides a useful categorization on which to base other ontologies.
- facilitates the integration of domain ontologies.

At the top is entity. OWL calls the top of the hierarchy thing. Essentially, everything is an entity.

Concrete or abstract

- Physical objects and events are concrete.
  E.g., A person, a lecture, the sending of an email.
- Mathematic objects and times are abstract.
  E.g., 17, set of all mammals on Earth, an email, a course

Continuants vs Occurrents

- A continuant exists in an instance of time and maintains its identity through time.
  Examples: person, a finger, a country, a smile, the smell of a flower, an email, Newtonian mechanics
- An occurrent has temporal parts.
  Examples: a life, a holoday, smiling, the opening of a flower, sending an email, earthquake
- Continuants participate in occurrents.
- a person, a life, a finger, infancy: what is part of what?

Alternative: a four-dimensional or perdurant view where objects exist in the space-time.

- A person is a trajectory though space and time
- At any time, a person is a snapshot of the four-dimensional trajectory.

Dependent or independent

- An independent continuant is something that can exist by itself or is part of another entity.
  For example, a person, a face, a pen, a flower, a country, and the atmosphere are independent continuants.
- A dependent continuant only exists by virtue of another entity and is not a part of that entity.
  For example, a smile, the ability to laugh, the inside of your mouth, the ownership relation between a person and a phone
- An occurrent dependent on an entity is a process or an event.
- A process happens over time, has temporal parts, and depends on a continuant.
  For example: a holiday, writing an email, and a robot cleaning the lab are all processes.
- An event is something that happens at an instant, and is often a process boundary.
  For example, the end of a lecture, the first goal in the 2022 FIFA World Cup final.

Provenance

- The provenance of data or data lineage specifies where the data came from and how it was manipulated
- Provenance is typically recorded as metadata – data about the data – including:
  - Who collected each piece of data? What are their credentials?
  - Who transcribed the information?
  - What was the protocol used to collect the data? Was the data chosen at random or chosen because it was interesting or some other reason?
  - What were the controls? What was manipulated, when?
  - What sensors were used? What is their reliability and operating range?
  - What processing has been done to the data?

FAIR data

FAIR principles for data:

- *Findable* – the (meta)data uses unique persistent identifiers, such as IRIs.
- *Accessible* – the data is available using free and open protocols, and the metadata is accessible even when the data is not.
- *Interoperable* – the vocabulary is defined using formal knowledge representation languages (ontologies).
- *Reusable* – the data uses rich metadata, including provenance, and an appropriate open license, so that the community can use the data.

Ontologies in Science

- https://schema.org
- SNOMED CT for medicine:
  https://www.snomed.org/five-step-briefing or
  https://browser.ihtsdotools.org/
- http://obofoundry.org
- https://www.springernature.com/gp/open-research/open-data