Chapter 13:

Ontologies

Ontologies

- Is there a flexible way to represent relations?
- How can knowledge bases be made to inter-operate semantically?

Choosing Individuals and Relations

How to represent: "Pen #7 is red."

Choosing Individuals and Relations

How to represent: "Pen #7 is red."

- *red*(pen_7). It's easy to ask "What's red?"
  Can't ask "what is the color of pen_7?"

Choosing Individuals and Relations

How to represent: "Pen #7 is red."

- $red(pen_7)$. It's easy to ask "What's red?"
  Can't ask "what is the color of pen_7?"
- $color(pen_7, red)$. It's easy to ask "What's red?"
  It's easy to ask "What is the color of pen_7?"
  Can't ask "What property of pen_7 has value red?"

Choosing Individuals and Relations

How to represent: "Pen #7 is red."

- $red(pen_7)$. It's easy to ask "What's red?"
  Can't ask "what is the color of pen_7?"
- $color(pen_7, red)$. It's easy to ask "What's red?"
  It's easy to ask "What is the color of pen_7?"
  Can't ask "What property of pen_7 has value red?"
- $prop(pen_7, color, red)$. It's easy to ask all these questions.

Choosing Individuals and Relations

How to represent: "Pen #7 is red."

- $red(pen_7)$. It's easy to ask "What's red?"
  Can't ask "what is the color of pen_7?"
- $color(pen_7, red)$. It's easy to ask "What's red?"
  It's easy to ask "What is the color of pen_7?"
  Can't ask "What property of pen_7 has value red?"
- $prop(pen_7, color, red)$. It's easy to ask all these questions.

$prop(Individual, Property, Value)$ is the only relation needed:
called individual-property-value representation
or triple representation

Universality of *prop*

To represent "a is a parcel"

Universality of *prop*

To represent "a is a parcel"

- $prop(a, type, parcel)$, where *type* is a special property
- $prop(a, parcel, true)$, where *parcel* is a Boolean property

Reification

- To represent $scheduled(cs422, 2, 1030, cc208)$. "section 2 of course $cs422$ is scheduled at 10:30 in room $cc208$."

Reification

- To represent $scheduled(cs422, 2, 1030, cc208)$. "section 2 of course $cs422$ is scheduled at 10:30 in room $cc208$."
- Let $b123$ name the booking:

$$prop(b123, course, cs422).$$
$$prop(b123, section, 2).$$
$$prop(b123, time, 1030).$$
$$prop(b123, room, cc208).$$

- We have reified the booking.
- Reify means: to make into an individual.
- What if we want to add the year?

Semantics Networks

When you only have one relation, *prop*, it can be omitted without loss of information.

Logic:

$$prop(Individual, Property, Value)$$

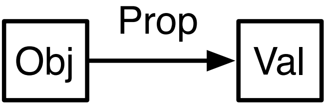
triple:

$$\langle Individual, Property, Value \rangle$$

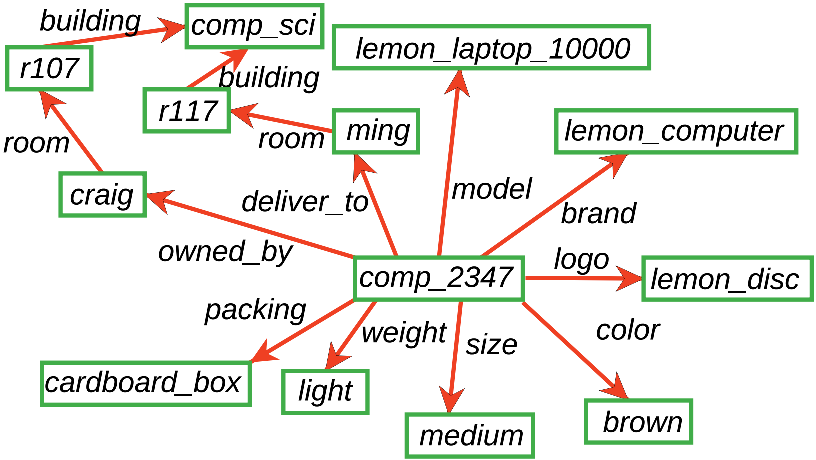
simple sentence:

$$Individual\ \ Property\ \ Value.$$

graphically:



An Example Semantic Network



Equivalent Logic Program

$prop(comp\_2347, owned\_by, craig).$

$prop(comp\_2347, deliver\_to, ming).$

$prop(comp\_2347, model, lemon\_laptop\_10000).$

$prop(comp\_2347, brand, lemon\_computer).$

$prop(comp\_2347, logo, lemon\_disc).$

$prop(comp\_2347, color, brown).$

$prop(craig, room, r107).$

$prop(r107, building, comp\_sci).$

$\vdots$

Turtle: a simple language of triples

A triple is written as

$$\textit{Subject Verb Object}.$$

A comma can group objects with the same subject and verb.

$S\ V\ O_1, O_2.$ is an abbreviation for $\begin{array}{l} S\ V\ O_1. \\ S\ V\ O_2. \end{array}$

A semi-colon can group verb-object pairs for the same subject.

$S\ V_1\ O_1; V_2\ O_2.$ is an abbreviation for $\begin{array}{l} S\ V_1\ O_1. \\ S\ V_2\ O_2. \end{array}$

Square brackets can be used to define an individual that is not given an identifier. It can then be used as the object of a triple.

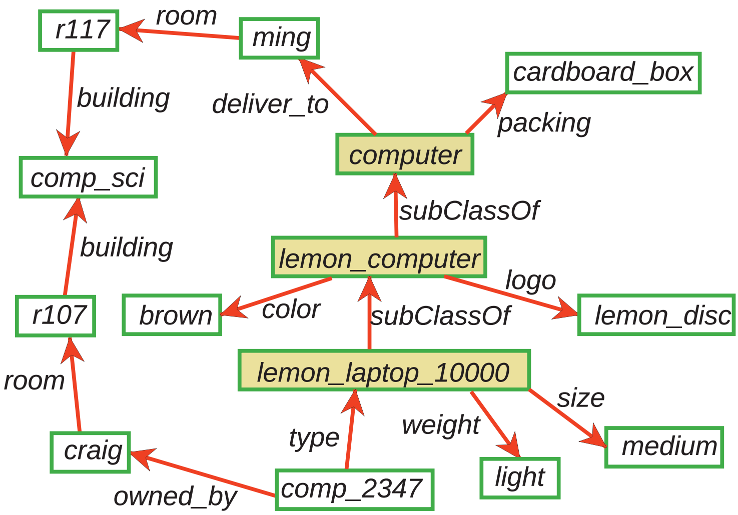
Turtle Example

```
⟨comp_3645⟩  ⟨#owned_by⟩  ⟨#fran⟩ ;
             ⟨#color⟩  ⟨#green⟩ , ⟨#yellow⟩ ;
             ⟨#managed_by⟩ [ ⟨#occupation⟩  ⟨#sys_admin⟩ ;
                             ⟨#serves_building⟩  ⟨#comp_sci⟩].
```

Primitive versus Derived Properties

- Primitive knowledge is that which is defined explicitly by facts.
- Derived knowledge is knowledge defined by rules.
- a class is a set of individuals that are grouped together as they have similar properties.
- Example: All lemon computers may have $color = brown$. Associate this property with the class, not the individual.
- Allow a special property *type* between an individual and a class.
- Use a special property *subClassOf* between two classes that allows for property inheritance.

A Structured Semantic Network



Logic of Property

An arc $c \xrightarrow{p} v$ from a class c with a property p to value v means every individual in the class has value v on property p:

$$prop(Obj, p, v) \leftarrow$$
$$\quad prop(Obj, type, c).$$

Example:

$$prop(X, weight, light) \leftarrow$$
$$\quad prop(X, type, lemon\_laptop\_10000).$$
$$prop(X, packing, cardboard\_box) \leftarrow$$
$$\quad prop(X, type, computer).$$

Logic of Property Inheritance

You can do inheritance through the subclass relationship:

$$
\begin{aligned}
& prop(X, type, T) \leftarrow \\
& \qquad prop(S, subClassOf, T) \wedge \\
& \qquad prop(X, type, S).
\end{aligned}
$$

Multiple Inheritance

- An individual is usually a member of more than one class. For example, the same person may be a wine expert, a teacher, a football coach,. . . .
- The individual can inherit the properties of all of the classes it is a member of: multiple inheritance.
- With default values,what if an individual inherits conflicting defaults from the different classes?
  multiple inheritance problem.

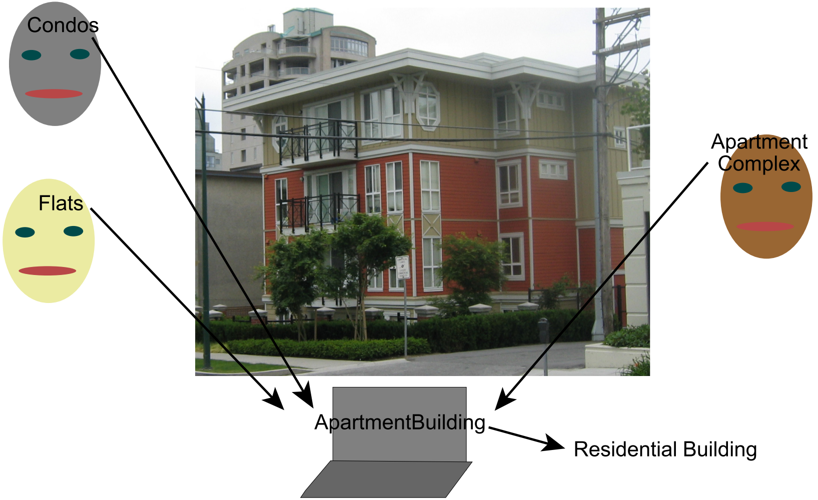
Choosing Primitive and Derived Properties

- Associate a property value with the most general class with that property value.
- Don't associate contingent properties of a class with the class. For example, if all of current computers just happen to be brown.

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.
- 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

- Ontologies can be huge: finding the appropriate terminology for a concept may be difficult.

Challenges of building ontologies

- Ontologies 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.

Challenges of building ontologies

- Ontologies 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.

Challenges of building ontologies

- Ontologies 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

- XML the Extensible Markup Language provides generic syntax.
  $\langle tag \ldots / \rangle$ or
  $\langle tag \ldots \rangle \ldots \langle / tag \rangle$.
- URI a Uniform Resource Identifier is a name of an individual (resource). This name can be shared. Often in the form of a URL to ensure uniqueness.
- RDF the Resource Description Framework is a language of triples
- OWL the Web Ontology Language, defines some primitive properties that can be used to define terminology. (Doesn't define a syntax).

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 relationships 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.
- The UNA is not an assumption we can universally make: "The Queen", "Elizabeth Windsor", etc.
- Without determining equality, we can't count!
- In OWL we can specify:

$$i_1 \;\; \mathit{SameIndividual} \;\; i_2.$$

$$i_1 \;\; \mathit{DifferentIndividuals} \;\; i_3.$$

Classes

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

  house subClassOf building.

  officeBuilding subClassOf building.
- The most general class is *Thing*.
- Classes can be declared to be the same or to be disjoint:

  house EquivalentClasses singleFamilyDwelling.

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

Properties

- A property is a relationship between an individual and a value.
- A property has a domain (for the individual) and a range (for the value).

  livesIn *domain* *person*.

  livesIn *range* *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:

  livesIn *subPropertyOf* *enclosure*.

  principalResidence *subPropertyOf* *livesIn*.

Properties (Cont.)

- One property can be inverse of another

  livesIn InverseObjectProperties 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.

  principalResidence minCardinality 1.

  principalResidence maxCardinality 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.

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.

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

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.

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

homeOwner subClassOf person.
homeOwner subClassOf
    ObjectSomeValuesFrom(owns, house).

OWL Class Constructors

owl:Thing $\equiv$ all individuals

owl:Nothing $\equiv$ no individuals

owl:ObjectIntersectionOf$(C_1, \ldots, C_k) \equiv C_1 \cap \cdots \cap C_k$

owl:ObjectUnionOf$(C_1, \ldots, C_k) \equiv C_1 \cup \cdots \cup C_k$

owl:ObjectComplementOf$(C) \equiv Thing \setminus C$

owl:ObjectOneOf$(I_1, \ldots, I_k) \equiv \{I_1, \ldots, I_k\}$

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

`numberOfUnits` is a property which maps residential buildings onto some values for the number of living units

Example: Apartment Building

`numberOfUnits` is a property which maps residential buildings onto some values for the number of living units

```
:numberOfUnits
    rdf:type      owl:FunctionalObjectProperty;
    rdfs:domain   :ResidentialBuilding;
    rdfs:range    owl:OneOf(:one :two :moreThanTwo).
```

Example: Apartment Building

`numberOfUnits` is a property which maps residential buildings onto some values for the number of living units

```
:numberOfUnits
    rdf:type       owl:FunctionalObjectProperty;
    rdfs:domain    :ResidentialBuilding;
    rdfs:range     owl:OneOf(:one :two :moreThanTwo).
```

`ownership` is a property which maps residential buildings onto values describing their legal status

Example: Apartment Building

`numberOfUnits` is a property which maps residential buildings onto some values for the number of living units

```
:numberOfUnits
    rdf:type       owl:FunctionalObjectProperty;
    rdfs:domain    :ResidentialBuilding;
    rdfs:range     owl:OneOf(:one :two :moreThanTwo).
```

`ownership` is a property which maps residential buildings onto values describing their legal status

```
:ownership
    rdf:type       owl:FunctionalObjectProperty;
    rdfs:domain    :ResidentialBuilding;
    rdfs:range     owl:OneOf(:rental :ownerOccupied :coop).
```

Example: Apartment Building

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

Example: Apartment Building

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

```
:ApartmentBuilding
   owl:EquivalentClasses
        owl:ObjectIntersectionOf (

             owl:ObjectHasValue(:numberOfUnits
                                :moreThanTwo)
             owl:ObjectHasValue(:onwership
                                :rental)
            :ResidentialBuilding).
```

Aristotelian definitions

Aristotle [350 B.C.] suggested the definition if a class C in terms of:

- Genus: the super-class
- Differentia: the attributes that make members of the class C different from other members of the super-class

"If genera are different and co-ordinate, their differentiae are themselves different in kind. Take as an instance the genus 'animal' and the genus 'knowledge'. 'With feet', 'two-footed', 'winged', 'aquatic', are differentiae of 'animal'; the species of knowledge are not distinguished by the same differentiae. One species of knowledge does not differ from another in being 'two-footed'."

Aristotle, *Categories*, 350 B.C.

Basic Formal Ontology (BFO)

- entity
  - continuant
    - independent continuant
      - site
      - object aggregate
      - object
      - fiat part of object
      - boundary of object
    - dependent continuant
      - realizable entity
        - function
        - role
        - disposition
      - quality
    - spatial region
      - volume / surface / line / point

BFO (cont.)

- occurrent
  - temporal region
    - connected temporal region
      - temporal interval
      - temporal instant
    - scattered temporal region
  - spatio-temporal region
    - connected spatio-temporal region
      - spatio-temporal interval / spatio-temporal instant
    - scattered spatio-temporal region
  - processual entity
    - process
    - process aggregate
    - processual context
    - fiat part of process
    - boundary of process

Continuants vs Occurrents

- A continuant exists in an instance of time and maintains its identity through time.
- An occurrent has temporal parts.
- Continuants participate in occurrents.
- a person, a life, a finger, infancy: what is part of what?

Continuants vs Occurrents

- A continuant exists in an instance of time and maintains its identity through time.
- An occurrent has temporal parts.
- Continuants participate in occurrents.
- a person, a life, a finger, infancy: what is part of what?
- a holiday, the end of a lecture, an email, the sending of an email, the equator, earthquake, a smile, a laugh, the smell of a flower

Continuants

- a pen, a person, Newtonian mechanics, the memory of a past event: objects
- a flock of birds, the students in CS422, a card collection: object aggregates
- a city, a room, a mouth, the hole of a doughnut: site
- the dangerous part of a city, part of Grouse Mountain with the best view: fiat part of an object.