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
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.
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.
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.
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

- **XML** the Extensible Markup Language provides generic syntax.
  \[ \langle \text{tag} \ldots \rangle \text{ or } \langle \text{tag} \ldots \rangle \ldots \langle /\text{tag} \rangle. \]

- **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**: between individuals and their values
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 the determining equality, we can’t count!

In OWL we can specify:

\[ i_1 \text{ SamelIndividual } i_2. \]
\[ i_1 \text{ DifferentIndividuals } i_3. \]
Classes

A class is a set of individuals. E.g., house, building, officeBuilding.

One class can be a subclass of another

\[ \text{house} \  \text{subClassOf} \  \text{building}. \]
\[ \text{officeBuilding} \  \text{subClassOf} \  \text{building}. \]

The most general class is \text{Thing}.

Classes can be declared to be the same or to be disjoint:

\[ \text{house} \  \text{EquivalentClasses} \  \text{singleFamilyDwelling}. \]
\[ \text{house} \  \text{DisjointClasses} \  \text{officeBuilding}. \]

Different classes are not necessarily disjoint. E.g., a building can be both a commercial building and a residential building.
A property is between an individual and a value.
A property has a domain and a range.

\[ \text{livesIn domain person.} \]
\[ \text{livesIn range placeOfResidence.} \]

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:

\[ \text{livesIn subPropertyOf enclosure.} \]
\[ \text{principalResidence subPropertyOf livesIn.} \]
One property can be inverse of another

\textit{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.

\textit{principalResidence minCardinality 1}.

\textit{principalResidence maxCardinality 1}.
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.
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.

\[
\text{homeOwner} \subseteq \text{person} \cap \{ x : \exists h \in \text{house} \text{ such that } x \text{ owns } h \}
\]
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.

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

\[ \text{homeOwner} \text{ subClassOf person}. \]
\[ \text{homeOwner} \text{ subClassOf ObjectSomeValuesFrom}(\text{owns, house}). \]
owl:Thing ≡ all individuals
owl:Nothing ≡ 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**

\[
\text{rdf:type}(I, C) \equiv I \in C \\
\text{rdfs:subClassOf}(C_1, C_2) \equiv C_1 \subseteq C_2 \\
\text{owl:EquivalentClasses}(C_1, C_2) \equiv C_1 \equiv C_2 \\
\text{owl:DisjointClasses}(C_1, C_2) \equiv C_1 \cap C_2 = \{\} \\
\text{rdfs:domain}(P, C) \equiv \text{if } xPy \text{ then } x \in C \\
\text{rdfs:range}(P, C) \equiv \text{if } xPy \text{ then } y \in C \\
\text{rdfs:subPropertyOf}(P_1, P_2) \equiv xP_1y \text{ implies } xP_2y \\
\text{owl:EquivalentObjectProperties}(P_1, P_2) \equiv xP_1y \text{ if and only if } xP_2y \\
\text{owl:DisjointObjectProperties}(P_1, P_2) \equiv xP_1y \text{ implies not } xP_2y \\
\text{owl:InverseObjectProperties}(P_1, P_2) \equiv xP_1y \text{ if and only if } yP_2x \\
\text{owl:SameIndividual}(I_1, \ldots, I_n) \equiv \forall j \forall k \ I_j = I_k \\
\text{owl:DifferentIndividuals}(I_1, \ldots, I_n) \equiv \forall j \forall k \ j \neq k \text{ implies } I_j \neq I_k \\
\text{owl:FunctionalObjectProperty}(P) \equiv \text{if } xPy_1 \text{ and } xPy_2 \text{ then } y_1 = y_2 \\
\text{owl:InverseFunctionalObjectProperty}(P) \equiv \\
\text{if } x_1Py \text{ and } x_2Py \text{ then } x_1 = x_2 \\
\text{owl:TransitiveObjectProperty}(P) \equiv \text{if } xPy \text{ and } yPz \text{ then } xPz \\
\text{owl:SymmetricObjectProperty} \equiv \text{if } xPy \text{ then } yPx
\]
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.
An apartment building is a residential building with more than two units and they are rented.
An apartment building is a residential building with more than two units and they are rented.

:numberOfUnits rdf:type owl:FunctionalObjectProperty;
   rdfs:domain :ResidentialBuilding;
   rdfs:range owl:OneOf(:one :two :moreThanTwo).
An apartment building is a residential building with more than two units and they are rented.

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

:ApartmetBuilding
   owl:EquivalentClasses
   owl:ObjectIntersectionOf ( 
      owl:ObjectHasValue(:numberOfUnits :moreThanTwo)
      owl:ObjectHasValue(:onwership :rental)
      :ResidentialBuilding).
```
Aristotelian definitions

Aristotle [350 B.C.] suggested the definition of 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.
Example: hotel ontology

Define the following:

- Room
- BathRoom
- StandardRoom - what is rented as a room in a hotel
- Suite
- RoomOnly
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
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

©D. Poole and A. Mackworth 2010 Artificial Intelligence, Lecture 13.2, Page 26
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
A **continuant** exists in an instance of time and maintains its identity through time.

An **occurrence** has temporal parts.

Continuants participate in occurrences.

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.