Knowledge Representation:
Ontology

Hutchatai Chanlekha

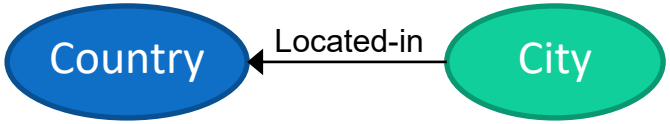
Department of Computer Engineering, Kasetsart University

Outline

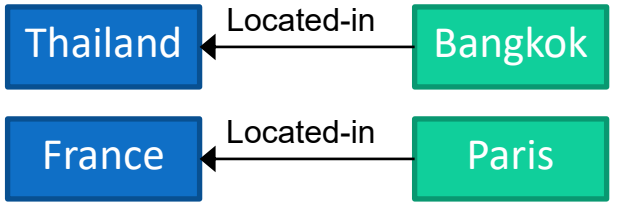
- Introduction to Ontology
- Class of Ontology

Ontology components

- Ontology components: **Concepts**, **properties**, and **instances**
  - Referenced by one or more symbols
  - Symbols are terms that humans can understand roughly by reading them
- These components are connected through relations
  - Semantic relations: link only concepts together



  - Instance relations: connect only instances
    - Instance relations are often instances of semantic relations
    - Relations between instances can be contextual and cannot be generalized to all instances of their concept



    - Example of contextual instance relation: person instance named “Mary Jane” is located in the city instance named “Bangkok” at the point in time “31 October 2015”
  - Terminological relations express the relationships that terms can have
    - Example: term “person” is synonym to the term “human being”

OWL ontology: Elements

- OWL: Web Ontology Language
- Most of the elements of an OWL ontology concern:
  - Classes
  - Properties (or Relations)
  - Instances of classes (i.e. individual)
  - Relationships between these instances.
- Distinction between a *class* and an *individual* in OWL

Class	Individual
• A name and collection of properties that describe a set of individuals	• The members of those classes • Correspond to actual entities that can be grouped into these classes

OWL ontology: Class VS Individual

- In building ontologies, class and individual distinction is frequently blurred:
  - ***Class vs. Instance:***
    - In certain contexts something that is obviously a class can itself be considered an instance of something else.
    - For example, in the wine ontology *CabernetSauvignonGrape*
      - Could be an individual that is an instance of Grape? OR a subclass of Grape?
        - *CabernetSauvingonGrape* is an instance of class Grape, denoting the actual grape varietal called Cabernet Sauvignon.
        - *CabernetSauvignonGrape* could be considered a class, the set of all actual Cabernet Sauvignon grapes.
- Note that the development of an ontology should be firmly driven by the intended usage.

OWL ontology: Property

- Property
  - *Properties* let us assert general facts about the concepts/classes and specific facts about individuals
  - Two types of property
    - *datatype properties*: classes to datatypes (or individuals to datatypes)
    - *object properties*: relate classes to classes (or individuals to individuals)

```
<owl:Class rdf:ID="VintageYear" />
  <owl:Class rdf:ID="Vintage“ />
   <owl:Class rdf:ID=“Wine“ />
```

```
<owl:ObjectProperty rdf:ID="vintageOf">
    <rdfs:domain rdf:resource="#Vintage" />
    <rdfs:range rdf:resource="#Wine" />
</owl:ObjectProperty>
```

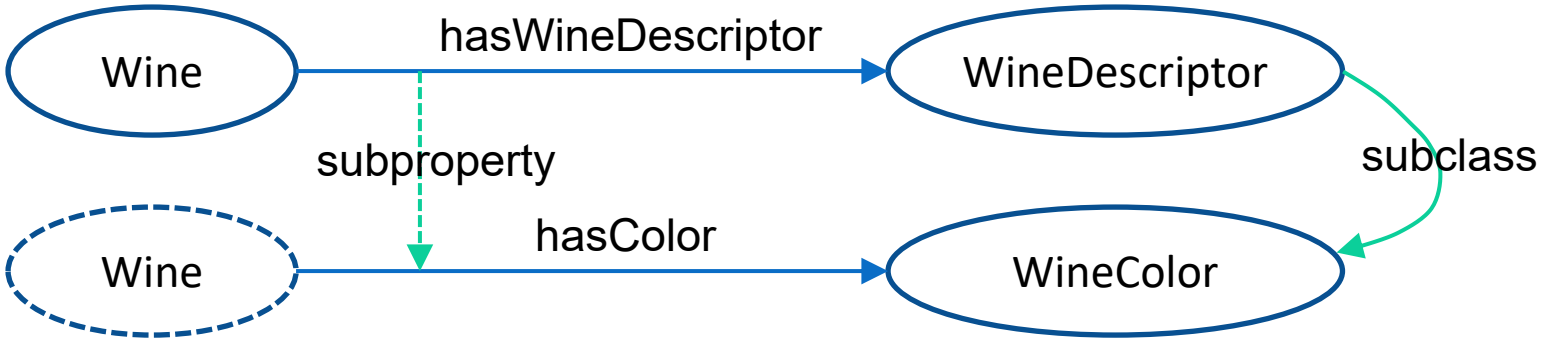
```
<owl:DatatypeProperty rdf:ID="yearValue">
     <rdfs:domain rdf:resource="#VintageYear" />
     <rdfs:range rdf:resource=“&xsd; positiveInteger"/>
</owl:DatatypeProperty>
```

OWL ontology: Property hierarchy

- Properties, like classes, can be arranged in a hierarchy.

```
<owl:ObjectProperty rdf:ID="hasWineDescriptor">
    <rdfs:domain rdf:resource="#Wine" />
    <rdfs:range rdf:resource="#WineDescriptor" />
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="hasColor">
    <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
    <rdfs:range rdf:resource="#WineColor" />
</owl:ObjectProperty>
```



OWL ontology: Property characteristics

- ***Property characteristics*** in OWL: Provides a powerful mechanism for enhanced reasoning about property

Property characteristics

- **TransitiveProperty**
  - If a property, P, is specified as transitive then for any x, y, and z:
    P(x,y) and P(y,z) implies P(x,z)
- **SymmetricProperty**
  - If a property, P, is tagged as symmetric then for any x and y:
    P(x,y) iff P(y,x)
- **FunctionalProperty**
  - If a property, P, is tagged as functional then for all x, y, and z:
    P(x,y) and P(x,z) implies y = z
- **inverseOf**
  - If a property, P1, is tagged as the owl:inverseOf P2, then for all x and y:
    P1(x,y) iff P2(y,x)
- **InverseFunctionalProperty**
  - If a property, P, is tagged as InverseFunctional then for all x, y and z:
    P(y,x) and P(z,x) implies y = z

OWL ontology: Property restrictions

- Property restrictions
  - it is possible to further constrain the range of a property in specific contexts
    - **allValuesFrom**
      - Example: For all wines, if they have makers, all the makers are wineries.
    - **someValuesFrom**
      - Example: For all RiceDishes, they have at least one ingredient that is a Rice.
    - **Cardinality**
      - Define cardinality constraints.
    - **hasValue**
      - Specify *particular* property values of a class

OWL ontology: Property of individual

- Property of individual

Define Region, Winery, and Year individuals to be used as property values

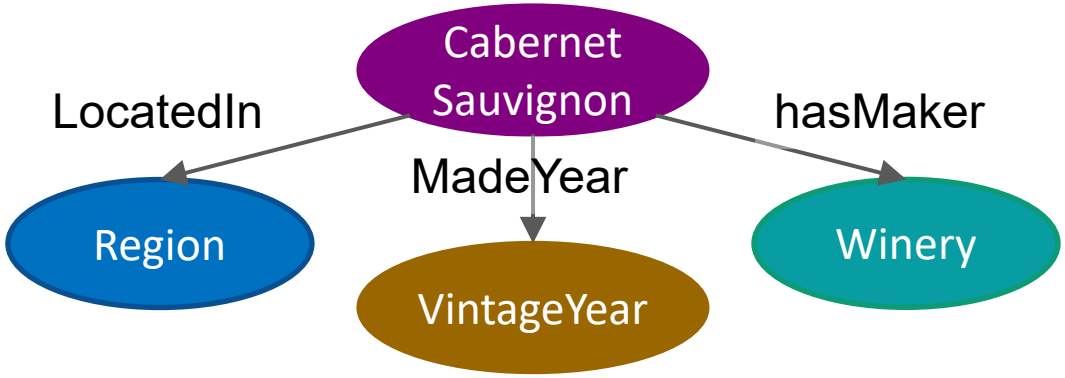
```
<Region rdf:ID="SantaCruzMountainsRegion">
    <locatedIn rdf:resource="#CaliforniaRegion" />
</Region>
<Winery rdf:ID="SantaCruzMountainVineyard" />
<VintageYear rdf:ID="Year1998">
    <yearValue rdf:datatype="&xsd;positiveInteger">1998</yearValue>
</VintageYear>
```

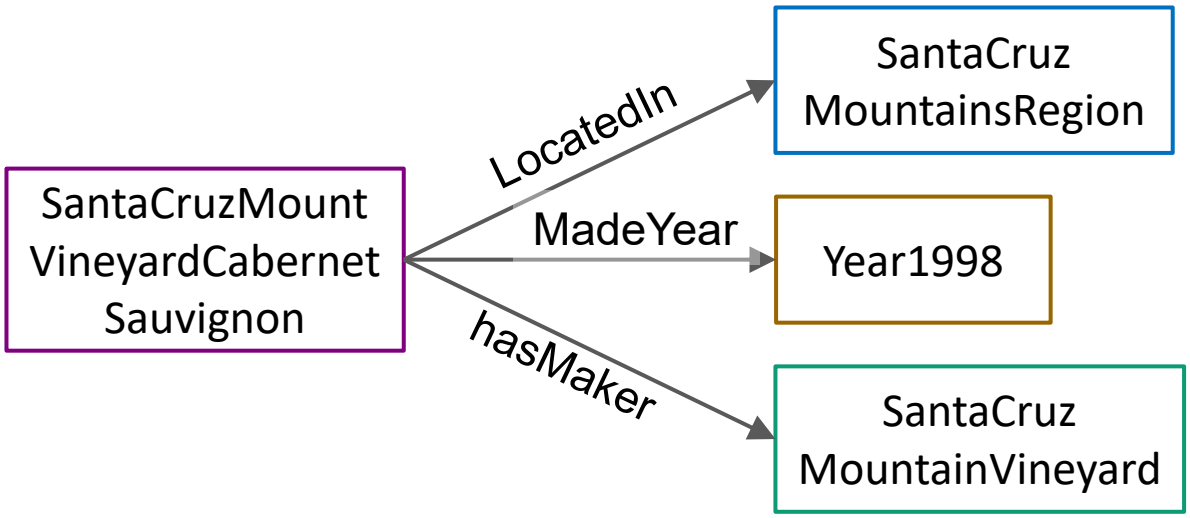
```
<CabernetSauvignon rdf:ID="SantaCruzMountVineyardCabernetSauvignon" >
    <locatedIn rdf:resource="#SantaCruzMountainsRegion"/>
    <hasMaker rdf:resource="#SantaCruzMountainVineyard" />
    <madeYear rdf:resource=#Year1998”>
</CabernetSauvignon>
```

Wine individual

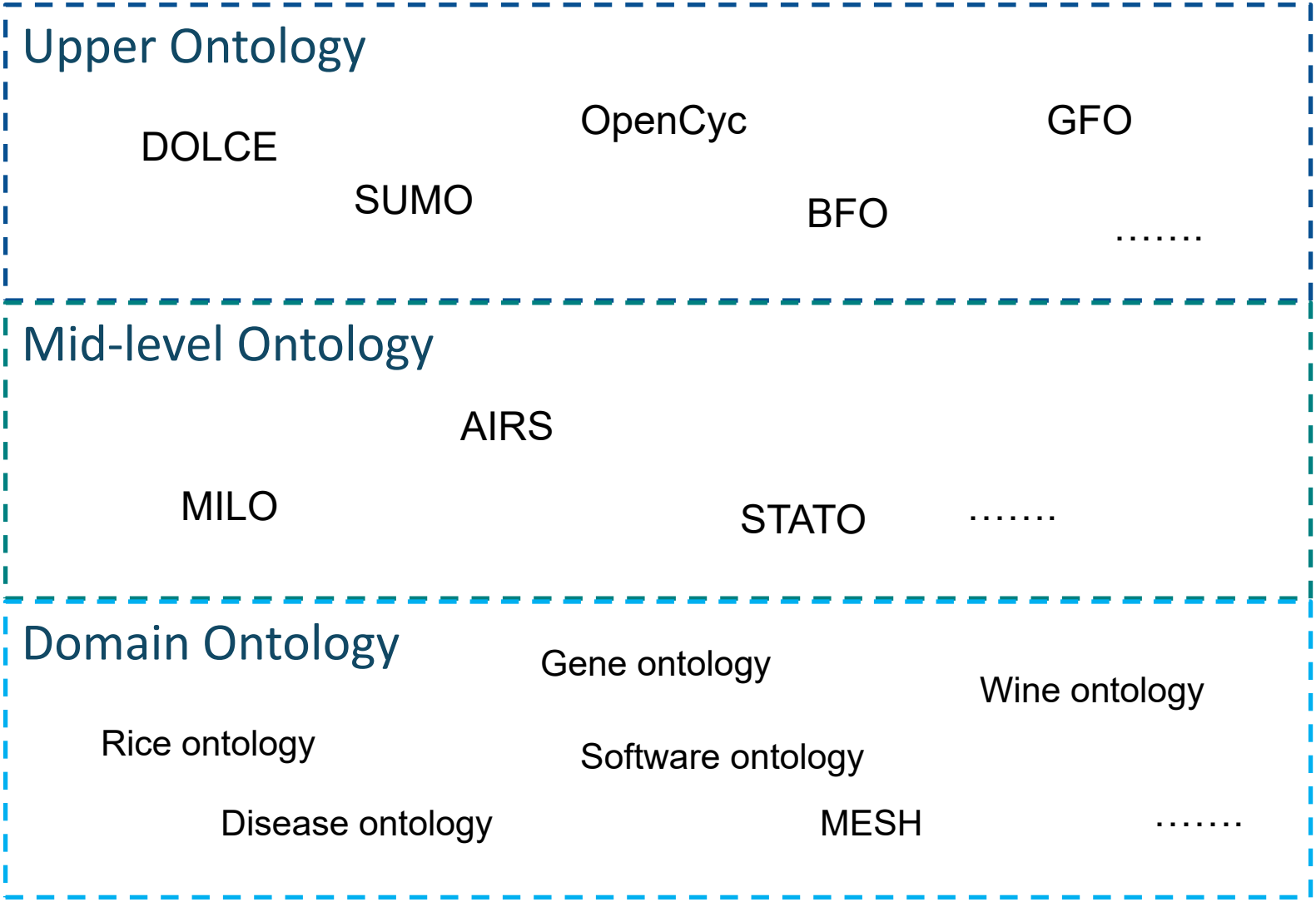
OWL ontology: Property of individual

- Property of individual





Broad classes of Ontologies



Upper Ontology

- Upper ontology or Foundational ontology or Universal ontology
  - Generic ontologies applicable to various domains
  - Define basic notions such as objects, relations, events, process, etc.
- An upper ontology is a high-level, domain-independent ontology
  - Provide a domain independent conceptual model that aims to be highly re-usable across specific domain applications
  - Often characterized as representing common sense concepts, i.e. those that are basic for human understanding of the world
- The concepts expressed in upper ontologies are intended to be basic and universal concepts
  - Usually describe very general level or abstract concepts
  - Ensure generality and expressivity for a wide area of domains.

Ref: http://ontology.ihmc.us/COI/Resources/UpperOntologyUseLong.htm

Upper Ontology (cont.)

- Provide a framework by which disparate systems may utilize
  - One of the primary purposes of upper ontologies is to aid semantic integration across ontologies and to encourage a set of design principles within those ontologies that use them
- Domain or core reference ontologies based on the same foundational ontology can be more easily integrated
- Can be used for deriving more domain-specific ontologies

- More information on upper ontologies
  - A Comparison of Upper Ontologies
    - http://www.disi.unige.it/person/MascardiV/Download/DISI-TR-06-21.pdf

Ref: http://ontology.ihmc.us/COI/Resources/UpperOntologyUseLong.htm

Mid-Level Ontology

- Serves as a bridge between abstract concepts defined in the upper ontology and low-level domain specific concepts specified in a domain ontology.
  - May provide more concrete representations of abstract concepts found in the upper ontology.
- Mid-level (as well as upper) ontologies are intended to provide a mechanism to make this mapping of concepts across domains easier.
- Represent knowledge at an intermediate level of detail independently of a specific task
- This ontology category also encompasses the set of ontologies that represent commonly used concepts, such as Time and Location.

Domain Ontology

- Specifies concepts particular to a domain of interest and represents those concepts and their relationships from a domain specific perspective.
- Domain ontology is only applicable to a domain with a specific view point.
  - View point defines how a group of users conceptualize and visualize some specific phenomenon
- Could be linked to a specific application

Domain Ontology

- Domain ontologies may be composed by importing mid-level ontologies. They may also extend concepts defined in mid-level or upper ontologies.
  - Reusing well-established ontologies in a domain ontology development allows one to take advantage of the semantic richness of the relevant concepts and logic already built into the reused ontology
  - Using common mid-level and upper ontologies is intended to ease the process of integrating or mapping domain ontologies.
- Ontologies designed for specific tasks are called "*application ontologies*"

Local Ontologies/Application Ontologies

- Specialization of domain ontologies
  - There could be No consensus or knowledge sharing
- Represents the particular model of a domain according to a single viewpoint of a user or a developer.

- Fonseca et al. (2000) present local ontology as a combination of domain ontology and task ontology in order to fulfill the specific purpose of an application.
  - Task ontology contains knowledge to achieve a task
  - Domain ontology describes the knowledge where the task is applied.

Ontology Engineering

- Ontology requirement specification
- Refinement
  - Knowledge elicitation and formalization
  - Produce a mature and application-oriented target ontology according to the specification
- Evaluation
  - Prove the usefulness of the developed ontology and the associated software environment
- Maintenance
  - Generally, things are constantly changing.
  - These changes must be reflected in the developed ontology, with the guarantee of coherence and compatible upgrade

Ontology Engineering [2]

- Reasoning
  - Ensure the quality of an ontology
  - Can be used in different phase of the ontology life cycle
    - During ontology design
      - Reasoning can be used to test whether concepts are non-contradictory, and to derive implied relations
    - During deployment
      - For determining the consistency of facts stated in annotations, or infer relationships between annotations instances and ontology classes
    - Interoperability and integration of different ontologies
      - After asserting some inter-ontology relationships, the integrated concept hierarchy is computed and the concepts are checked for consistency.

Ontology design approaches

Bottom-Up approach	Top-Down approach	Middle-Out approach
• Start from the most specific concepts • Build a structure by generalization • Prone to provide tailored and specific ontologies with fine detail grain concepts	• Start from the most generic concept • Build a structure by specialization • Top concepts can be chosen in a foundational ontology • Prone to reuse of ontologies and inclusion of high level philosophical considerations	• Identify central concepts in each area/domain • Core concept are identified and then generalized and specialized to complete the ontology

Reference

- Slides based on
  - **An Introduction to Ontologies and Ontology Engineering**
    - Catherine Roussey, Francois Pinet, Myoung Ah Kang, Oscar Corcho
    - http://oa.upm.es/10381/1/An_Introduction.pdf
  - **W3C OWL guide**
    - http://www.w3.org/TR/owl-guide/
  - http://www.slideshare.net/marinasantini1/09-semantic-webontologies