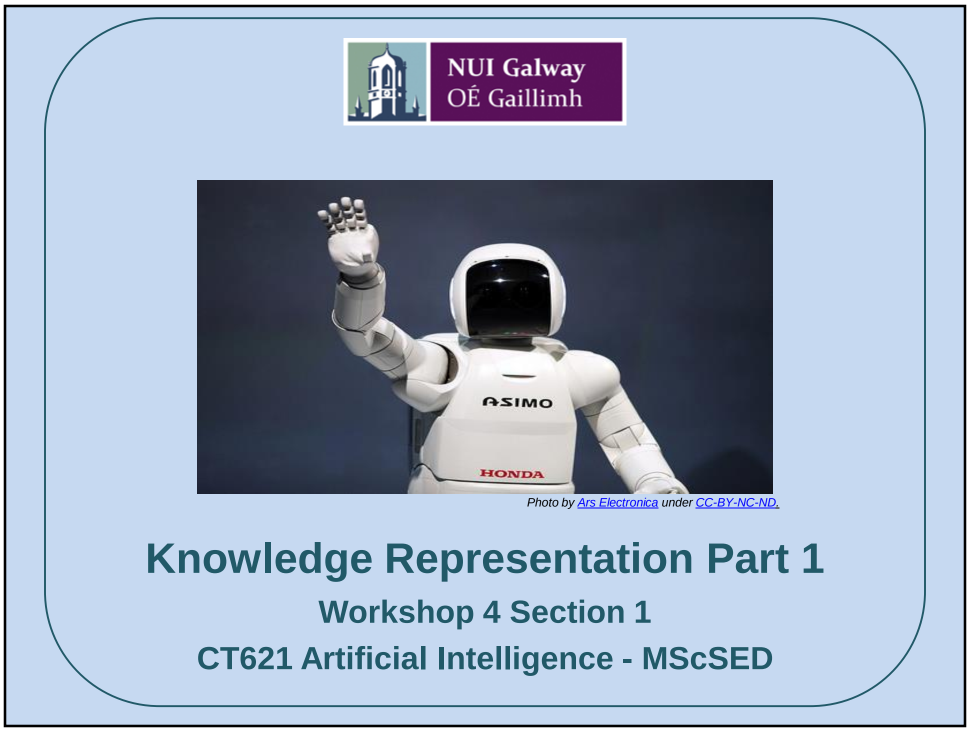


NUI Galway
OÉ Gaillimh

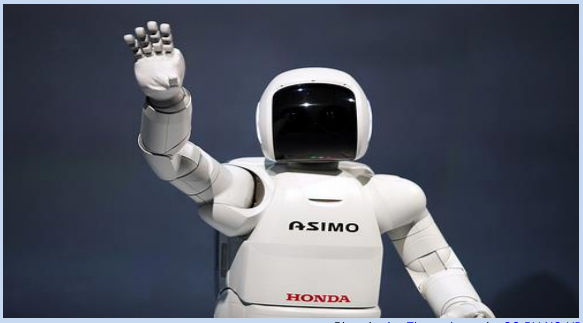


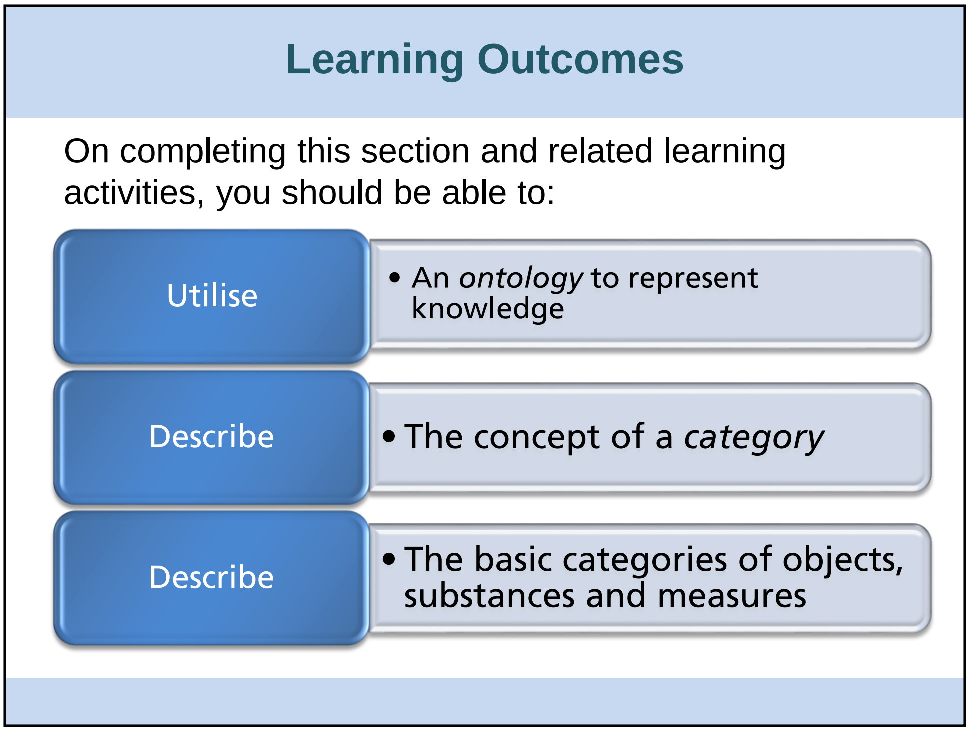
Photo by Ars Electronica under CC-BY-NC-ND.

Knowledge Representation Part 1

Workshop 4 Section 1

CT621 Artificial Intelligence - MScSED

In previous workshops, you have learnt about the theory of propositional and predicate logic, which provided us with a formal language for reasoning about knowledge. In the week 4 workshop we build on predicate logic in order to represent facts about the world. This enables us to create programs that reason about various aspects of the real world.

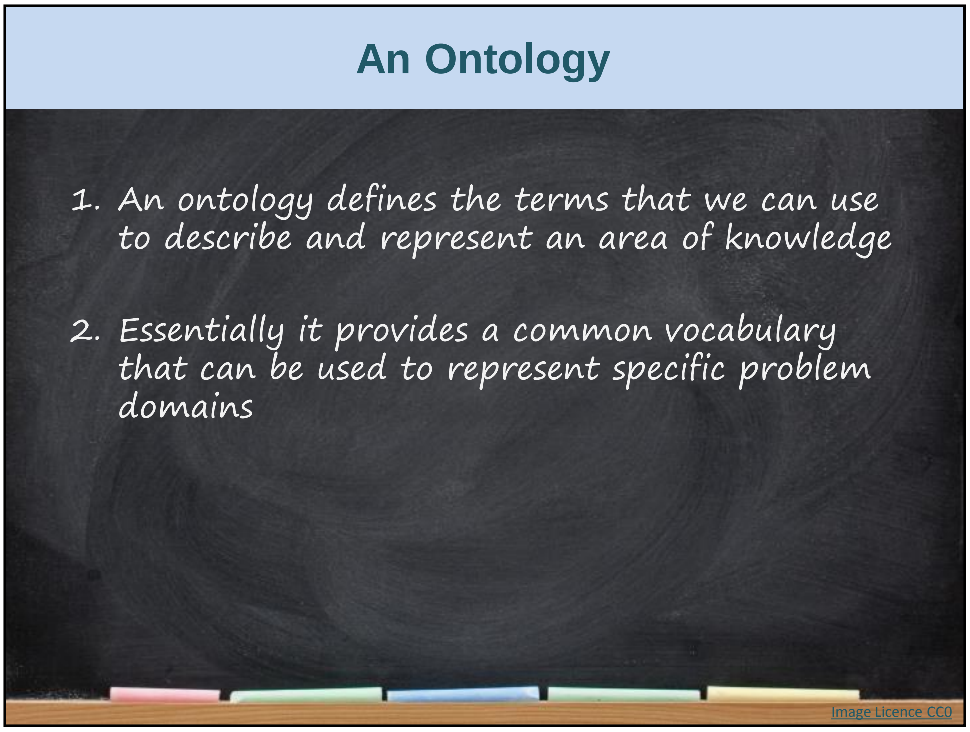


Learning Outcomes

On completing this section and related learning activities, you should be able to:

Utilise	• An *ontology* to represent knowledge
Describe	• The concept of a *category*
Describe	• The basic categories of objects, substances and measures

This workshop will begin by introducing the concept of an *ontology*, which can be used to organize everything in the world into a hierarchy of *categories*. We examine the basic categories of objects, substances and measures. The workshop then focuses on situation calculus and how it can be used to represent dynamic domains.



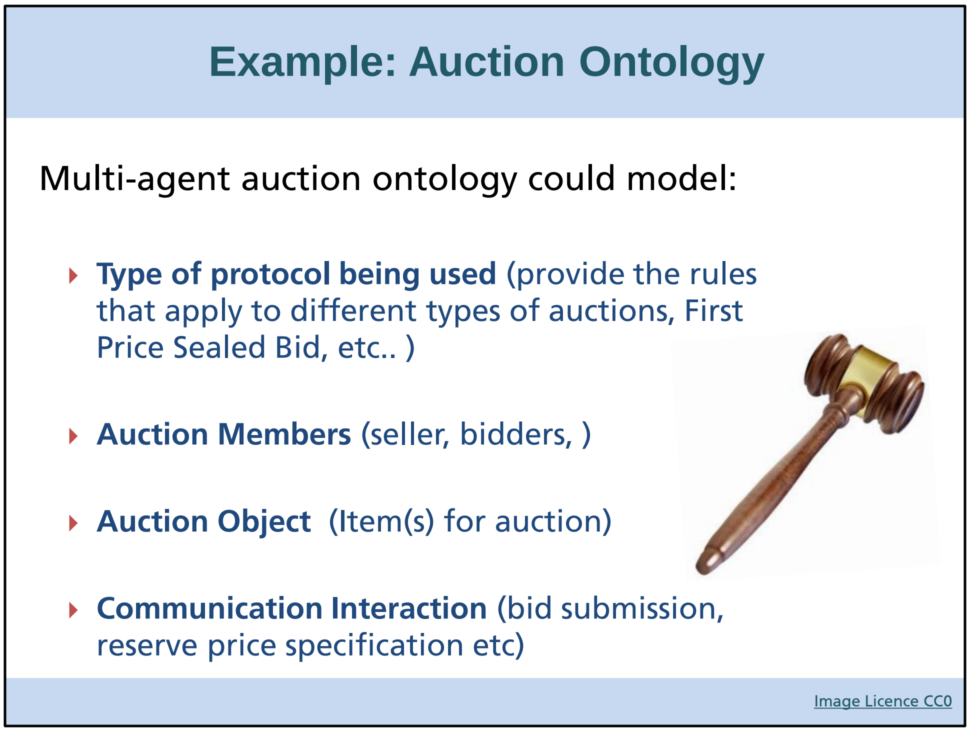
An Ontology

1. An ontology defines the terms that we can use to describe and represent an area of knowledge
2. Essentially it provides a common vocabulary that can be used to represent specific problem domains

Image Licence CC0

A key step in reasoning and representing knowledge is deciding on a vocabulary or ontology to describe facts about a particular domain. Frequently in AI, expert systems are created to tackle a specific problem, such as circuit design, medical diagnosis or the determination of a client's credit-worthiness. In designing an ontology for these systems, we define the predicates, functions and constants that are used to describe our knowledge about the domain.

Remember we defined the terms constant, predicate and function in Workshop1 Section 2. A constant is an object or property that does not change such as **Joe** and **Liam**. A function performs a mapping from one domain to another. Therefore if Joe is Liam's father then the function **father(Liam)** would return **Joe**. A predicate can be used to make assertions about objects and their relationships and will return either true or false. Therefore, if father was a predicate then **father(Liam)** would return true.



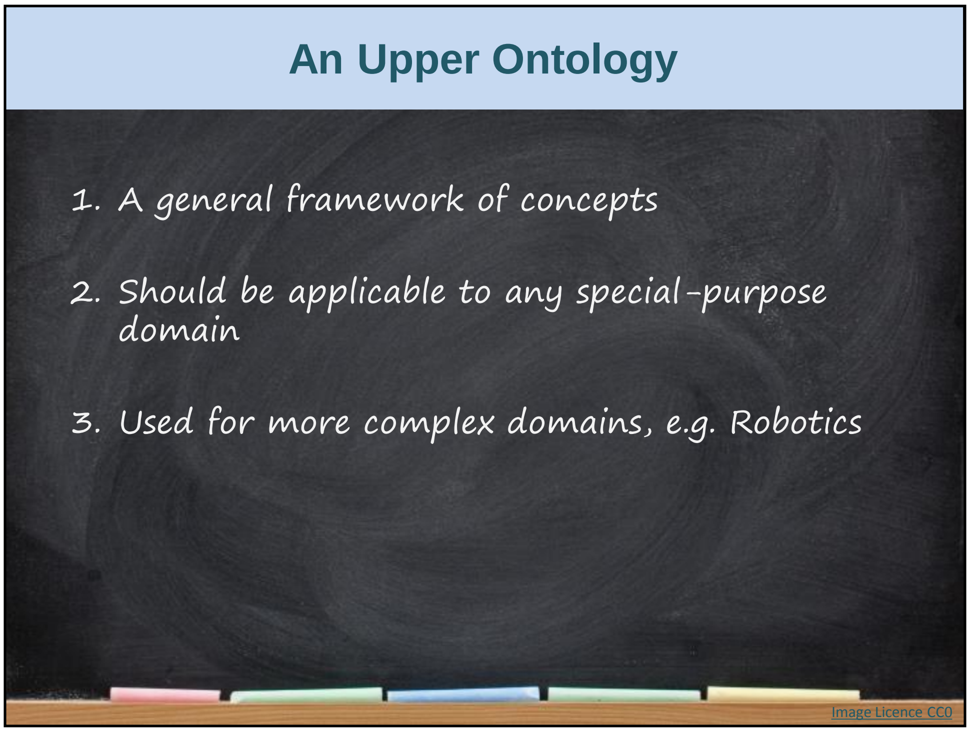
Example: Auction Ontology

Multi-agent auction ontology could model:

- **Type of protocol being used** (provide the rules that apply to different types of auctions, First Price Sealed Bid, etc..)
- **Auction Members** (seller, bidders,)
- **Auction Object** (Item(s) for auction)
- **Communication Interaction** (bid submission, reserve price specification etc)

An auction-based ontology would provide a formal description of both negotiation domain and the vocabulary used to facilitate that negotiation. Building an ontology is typically achieved by identifying the relevant objects within the domain and specifying how these objects are related or interact. For example, our auction ontology could model popular types of auction protocols such as the English auction, the first price sealed bid auction, etc. It could be used to define a bid, define participants such as bidders and sellers and the relationships and interaction between these various objects.

An ontology such as an auction ontology essentially represents a common view that can be used to model the domain and allows users to interact with the domain.



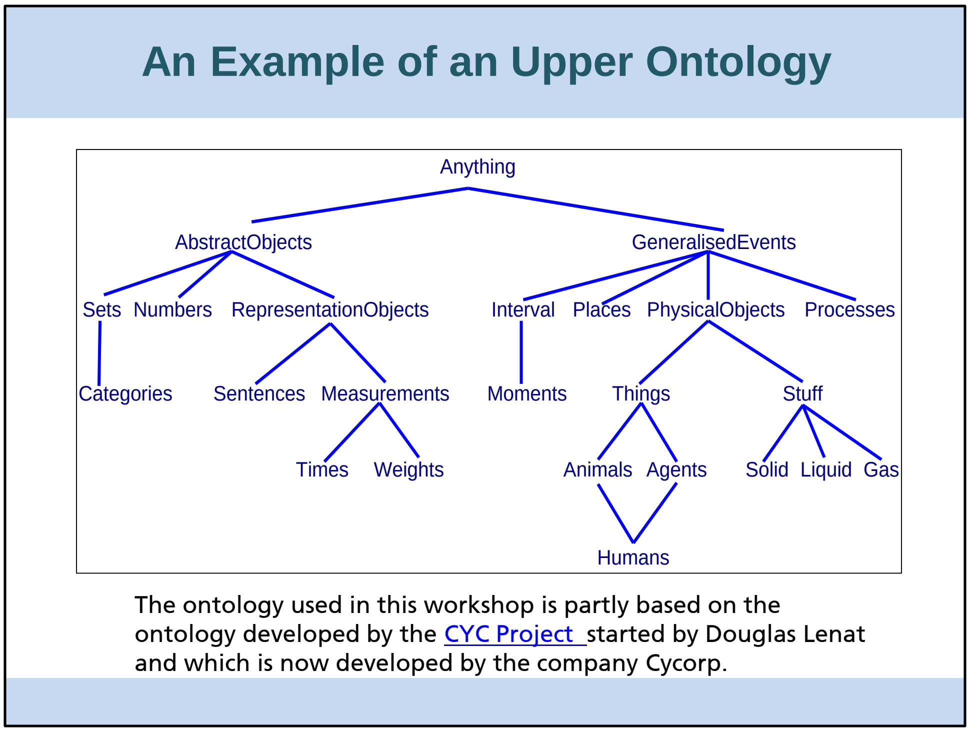
An Upper Ontology

1. A general framework of concepts
2. Should be applicable to any special-purpose domain
3. Used for more complex domains, e.g. Robotics

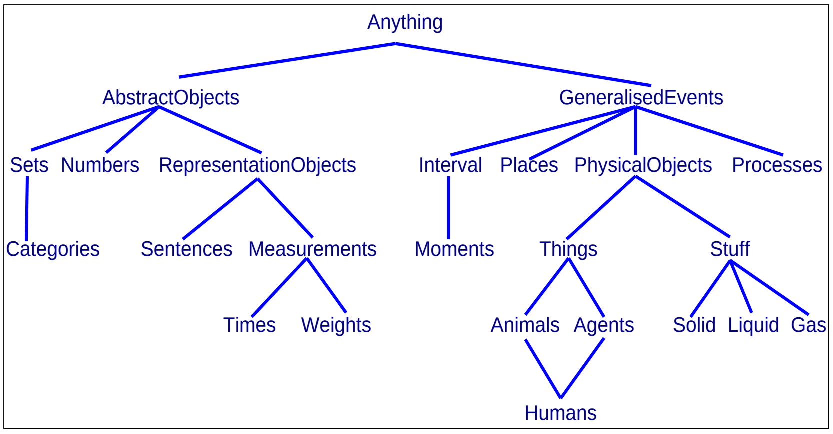
Image Licence CC0

Some complex large-scale problems, such as robot control, require a more flexible representation than that provided by domain-specific ontologies. A general-purpose ontology is known as an *upper ontology*. An upper ontology represents abstract concepts that occur in all domains.

It should be possible to apply an upper ontology to a specific domain through the addition of domain-specific axioms, i.e. the general framework should not have to be changed. Another key requirement of an upper ontology is that it should be capable of representing different areas of knowledge, e.g. should be able to describe a circuit in terms of its physical layout or in terms of its timing behaviour.

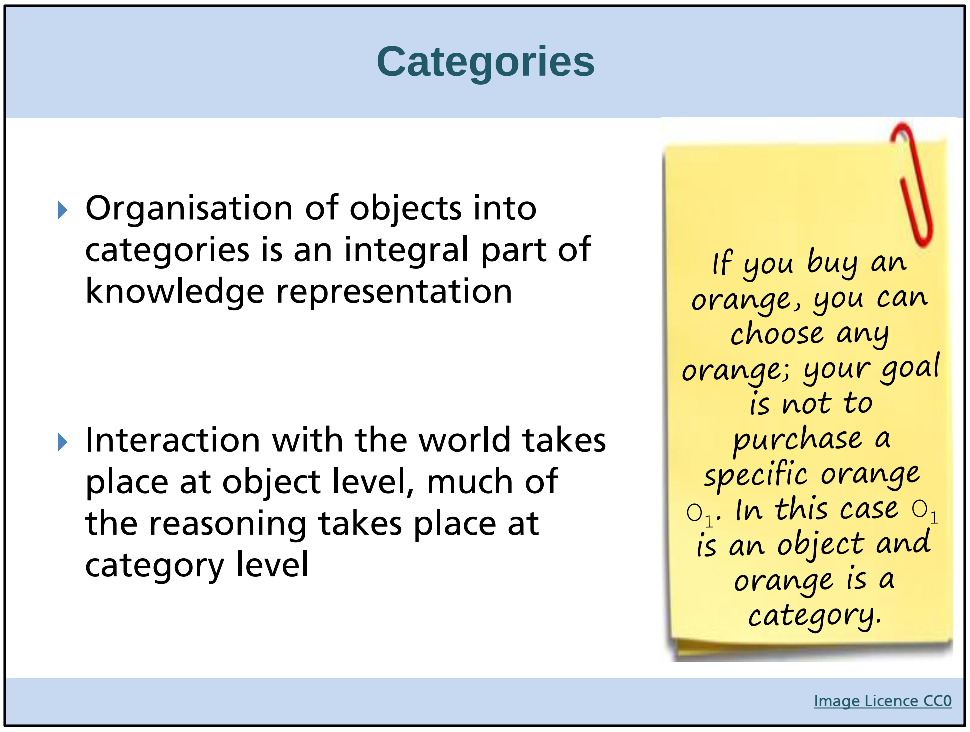


An Example of an Upper Ontology



The ontology used in this workshop is partly based on the ontology developed by the CYC Project started by Douglas Lenat and which is now developed by the company Cycorp.

This slide shows a diagram of an upper ontology. The more general concepts are shown at the top of the graph and the concepts become more specific further down the graph. As mentioned in the slide the ontology used in this workshop is partly based on the ontology used by the CYC project. The CYC project has produced a 6,000-concept upper ontology with 60,000 facts. An open-source version of the system is also available. The workshop ontology is also based on First-Order logic covered in earlier workshops.



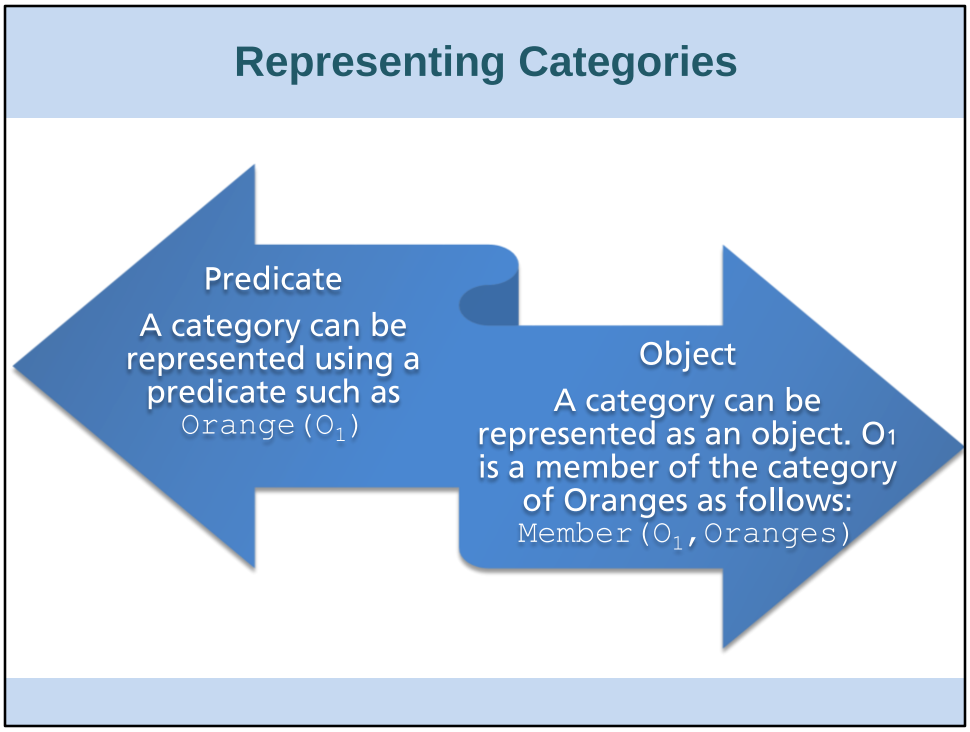
Categories

- Organisation of objects into categories is an integral part of knowledge representation
- Interaction with the world takes place at object level, much of the reasoning takes place at category level

If you buy an orange, you can choose any orange; your goal is not to purchase a specific orange O_1. In this case O_1 is an object and orange is a category.

Image Licence CC0

The definition of categories is a very important part of knowledge representation as much reasoning takes place at the level of categories. In the slide we make the distinction between a specific orange O_1 which is an object and the category `orange`. The orange O_1 can be seen as an instance of this category.



Representing Categories

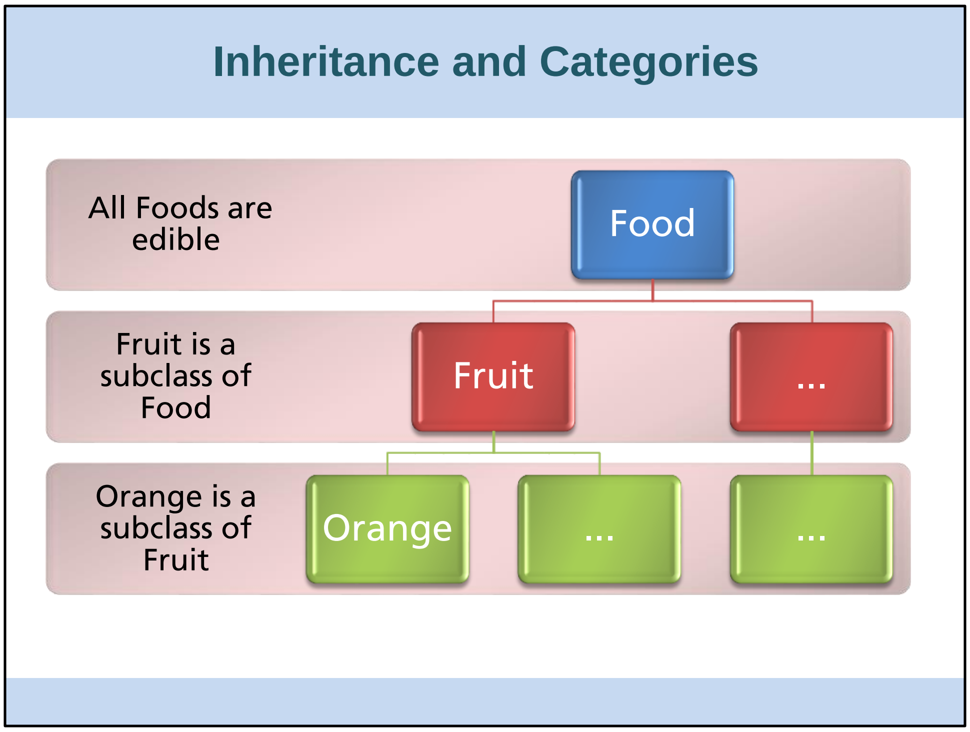
Predicate

A category can be represented using a predicate such as `Orange(O`$_1$`)`

Object

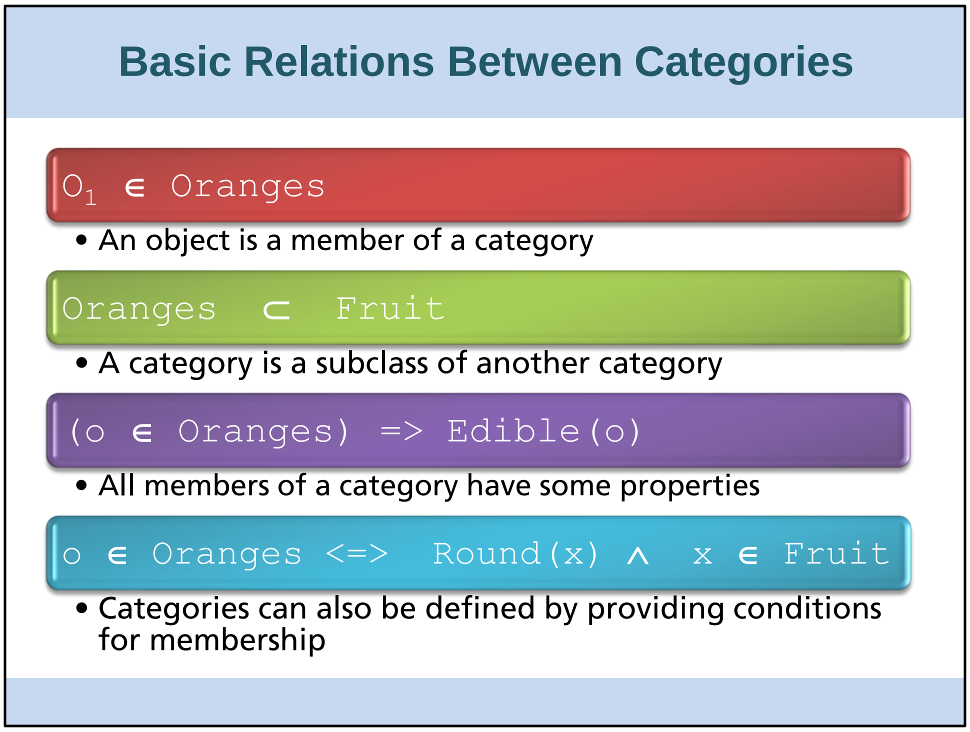
A category can be represented as an object. O_1 is a member of the category of Oranges as follows: `Member(O`$_1$`,Oranges)`

When using First Order Logic, there are two ways of representing categories: one uses predicates, e.g. `Orange(O`$_1$`)`. As we know a predicate returns true or false. The other can represent a category as an object and we can use a function such as `Member` to state that O_1 is a member of the Oranges category – `Member(O`$_1$`, Oranges)`. This can be abbreviated to $O_1 \in$ `Oranges`.



Inheritance and Categories

Categories can be used to effectively organize and simplify the representation of knowledge through inheritance. Consider the case where we state that all instances of the category `Food` are edible. If we further assert that `Fruit` is a subclass of food and `Orange` is a subclass of Fruit then we can infer that every orange is edible. In this case individual oranges inherit the attribute of edibility.



Basic Relations Between Categories

$O_1 \in \text{Oranges}$

- An object is a member of a category

$\text{Oranges} \subset \text{Fruit}$

- A category is a subclass of another category

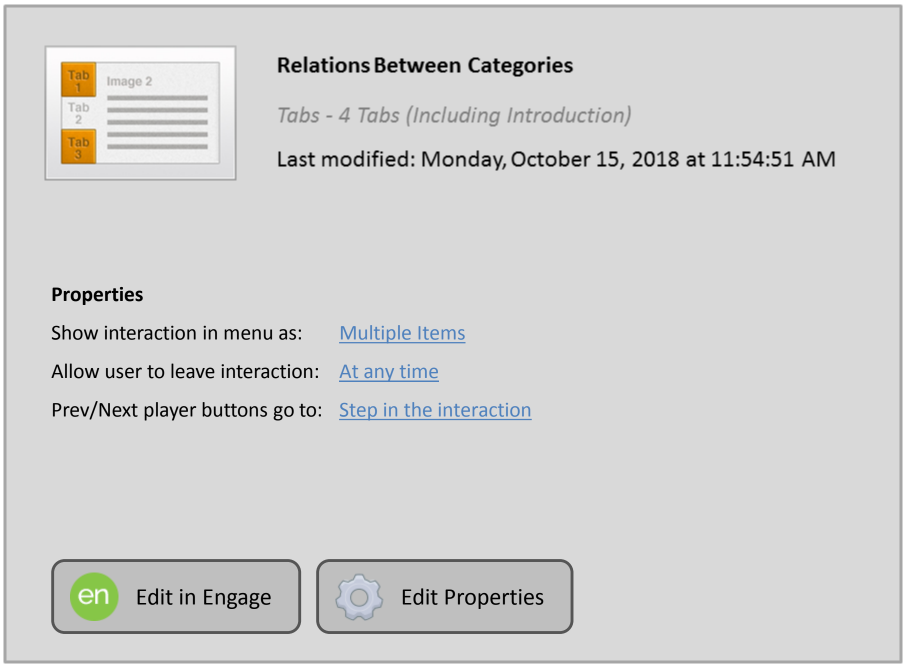
$(o \in \text{Oranges}) \Rightarrow \text{Edible}(o)$

- All members of a category have some properties

$o \in \text{Oranges} \Leftrightarrow \text{Round}(x) \wedge x \in \text{Fruit}$

- Categories can also be defined by providing conditions for membership

It is possible to state a basic fact or relationships about categories, either by relating objects to categories or by quantifying over their members.



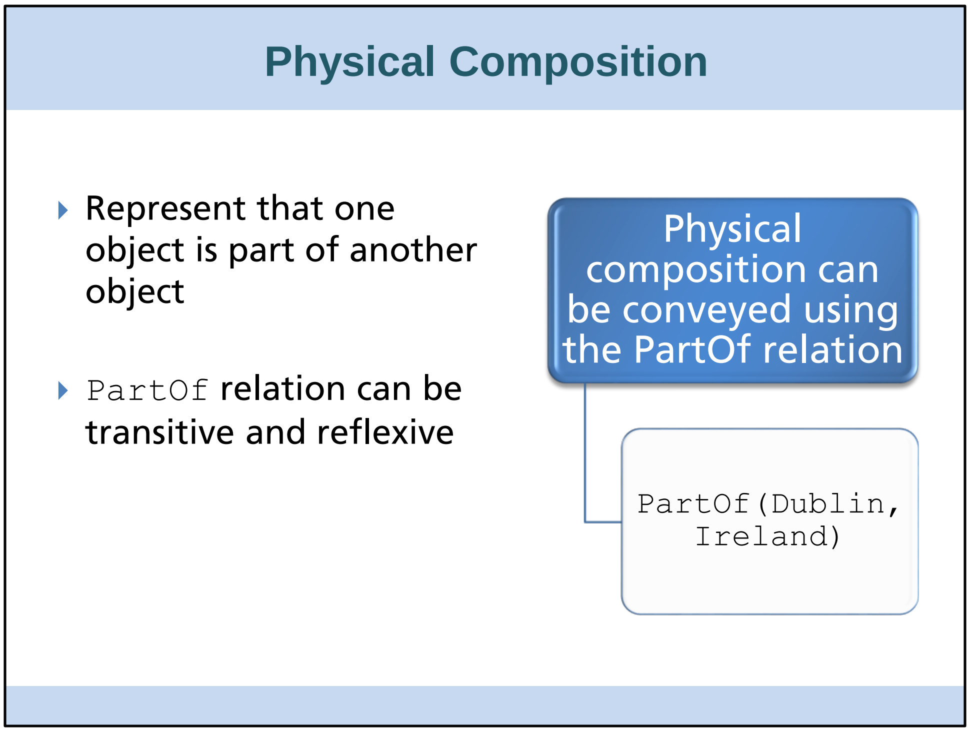
Relations Between Categories

Tabs - 4 Tabs (Including Introduction)

Last modified: Monday, October 15, 2018 at 11:54:51 AM

Properties

Show interaction in menu as:	Multiple Items
Allow user to leave interaction:	At any time
Prev/Next player buttons go to:	Step in the interaction



Physical Composition

- Represent that one object is part of another object
- `PartOf` relation can be transitive and reflexive

Physical composition can be conveyed using the PartOf relation
PartOf(Dublin, Ireland)

Physical composition is the straight-forward idea that one object is part of another object. For example, an engine is part of a car and a CPU is part of a computer. We can represent physical composition using the `PartOf` relation. See the following examples:

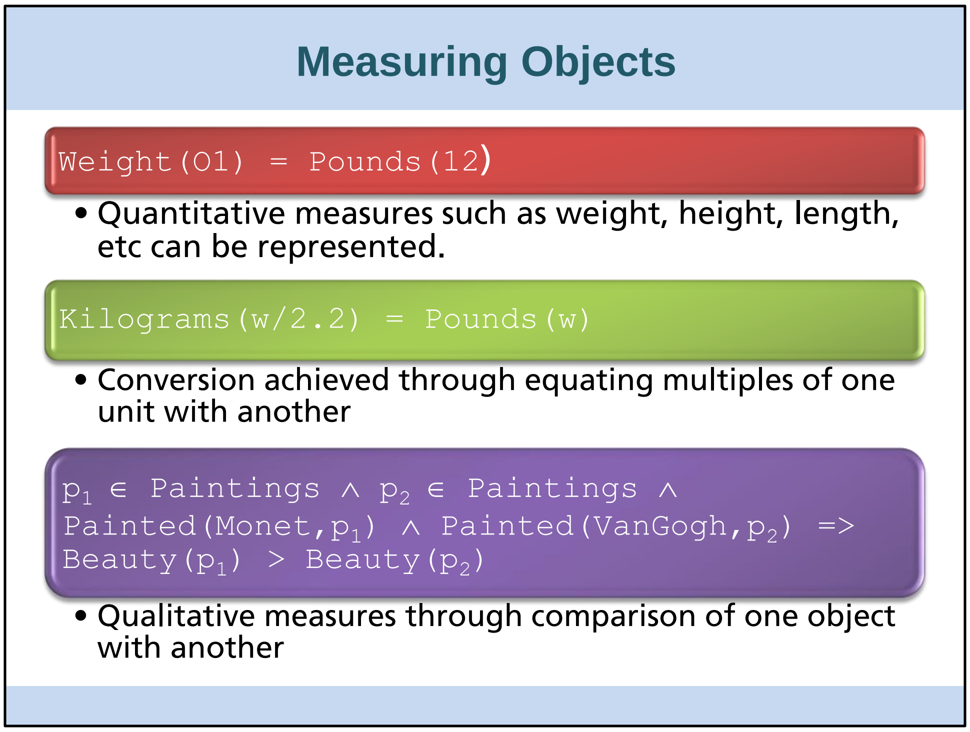
```
PartOf(engine, car)
PartOf(CPU, computer)
```

The `PartOf` relation is transitive. Therefore, if we know a piston is part of an engine and an engine is part of a car then we also know that a piston is part of a car.

```
PartOf(piston, engine) ^ PartOf(engine, car) =>
PartOf(piston, car)
```

The PartOf relation is reflexive which means that any object is part of itself
`PartOf(x, x)`

```
PartOf(engine, engine)
```



Measuring Objects

`Weight(O1) = Pounds(12)`

- Quantitative measures such as weight, height, length, etc can be represented.

`Kilograms(w/2.2) = Pounds(w)`

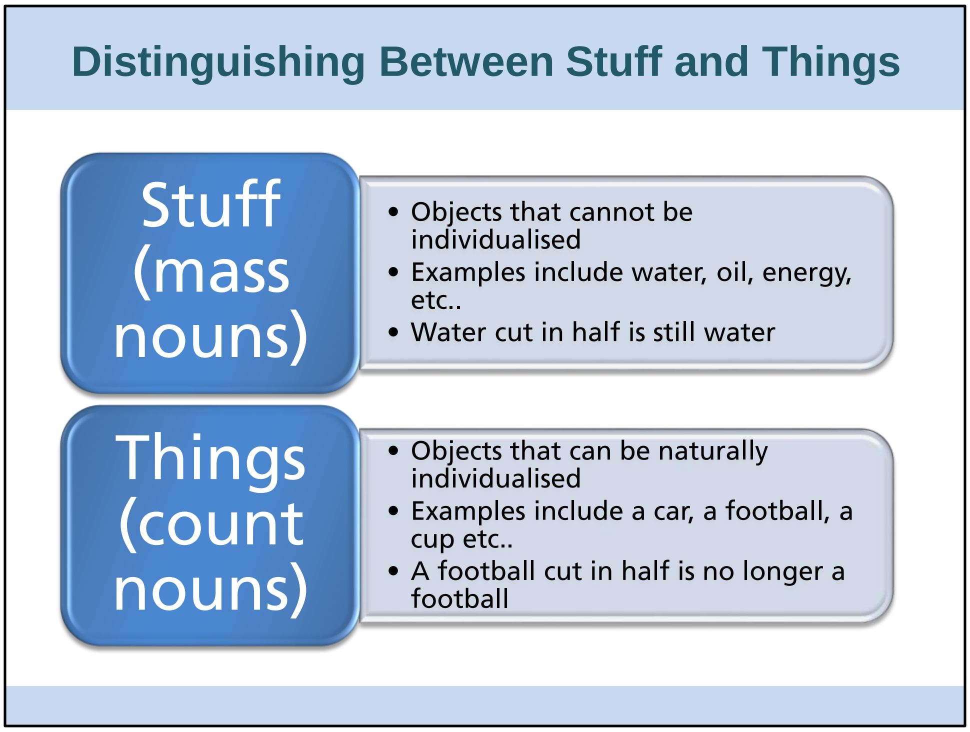
- Conversion achieved through equating multiples of one unit with another

$p_1 \in \text{Paintings} \wedge p_2 \in \text{Paintings} \wedge \text{Painted(Monet}, p_1) \wedge \text{Painted(VanGogh}, p_2) \Rightarrow \text{Beauty}(p_1) > \text{Beauty}(p_2)$

- Qualitative measures through comparison of one object with another

In describing objects of the real world, we often want to describe measurements of objects, such as height, weight, length, etc. One way to do this is to use a unit function with a number as in the example here which defines the weight of object O_1 as 12 pounds. To convert between different units, equate multiples of one unit to another. For example, mass in kilograms divided by 2.2 is equal to the mass in pounds.

Measurements do not have to have a numeric value attached to them. The most important property of measures is that they are ordered. Consider, for example, measuring the beauty of a painting or the funniness of a film – these are qualitative measures. In the example here, we state that all paintings of Monet are more beautiful than the paintings of Van Gogh. You may of course have a different opinion.



Distinguishing Between Stuff and Things

Stuff (mass nouns)

- Objects that cannot be individualised
- Examples include water, oil, energy, etc..
- Water cut in half is still water

Things (count nouns)

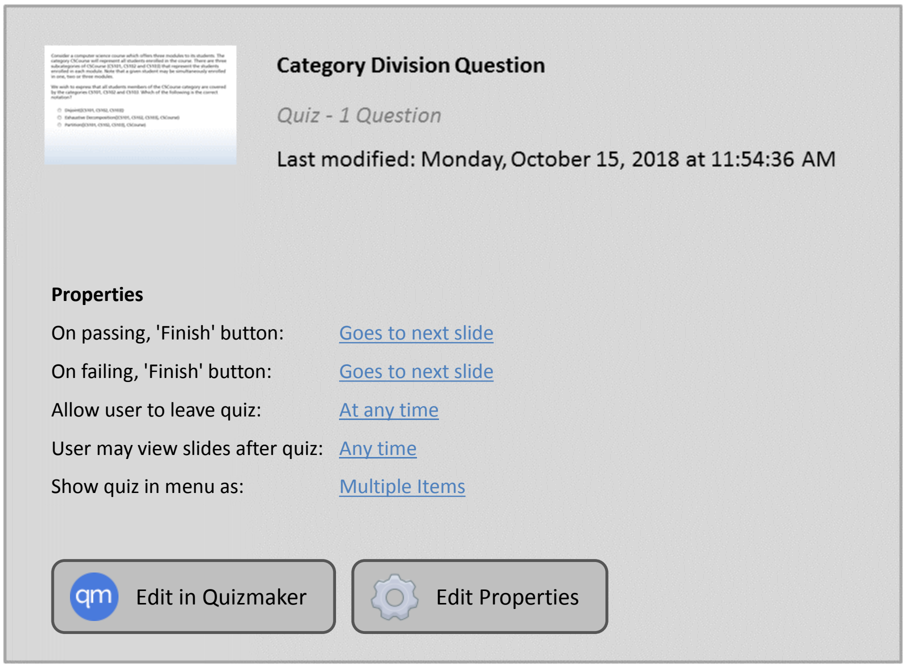
- Objects that can be naturally individualised
- Examples include a car, a football, a cup etc..
- A football cut in half is no longer a football

The ontology so far has dealt with countable objects or things, but not with stuff. Note that the Upper Ontology shown in an earlier slide divides the PhysicalObjects category into two sub-categories: **Things** and **Stuff**. Consider the difference between an aardvark and some butter. It is possible to count the number of aardvarks, but not the number of "butter-objects". Our ontology must therefore be able to distinguish between Things and Stuff, so that we can reason about them.

One major difference between Things and Stuff is that with butter-objects, for example, any part of a butter-object is also another butter-object.

```
b ∈ Butter ∧ PartOf(p,b)   => p ∈ Butter
```

However, any part of an aardvark (e.g. a leg) is not another aardvark.
Things and Stuff can be differentiated based on intrinsic and extrinsic properties. An **intrinsic** property is one that belongs to the substance of an object, so that when the object is divided into parts, each part retains that property. For example, the colour of a butter object. An **extrinsic** property is one that belongs to the object as a whole, e.g. the weight of an object.
A class of objects that includes in its definition only intrinsic properties is a mass noun. A class of objects that includes at least one extrinsic property is a count noun. All physical entities belong to both the generic Stuff and Thing category.



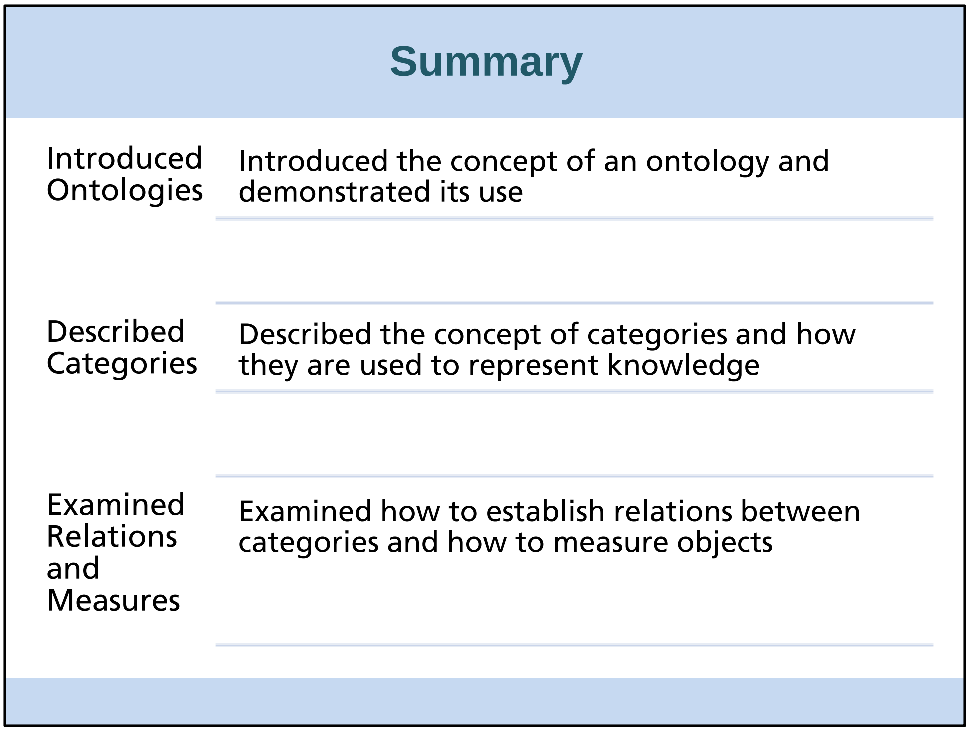
Category Division Question

Quiz - 1 Question

Last modified: Monday, October 15, 2018 at 11:54:36 AM

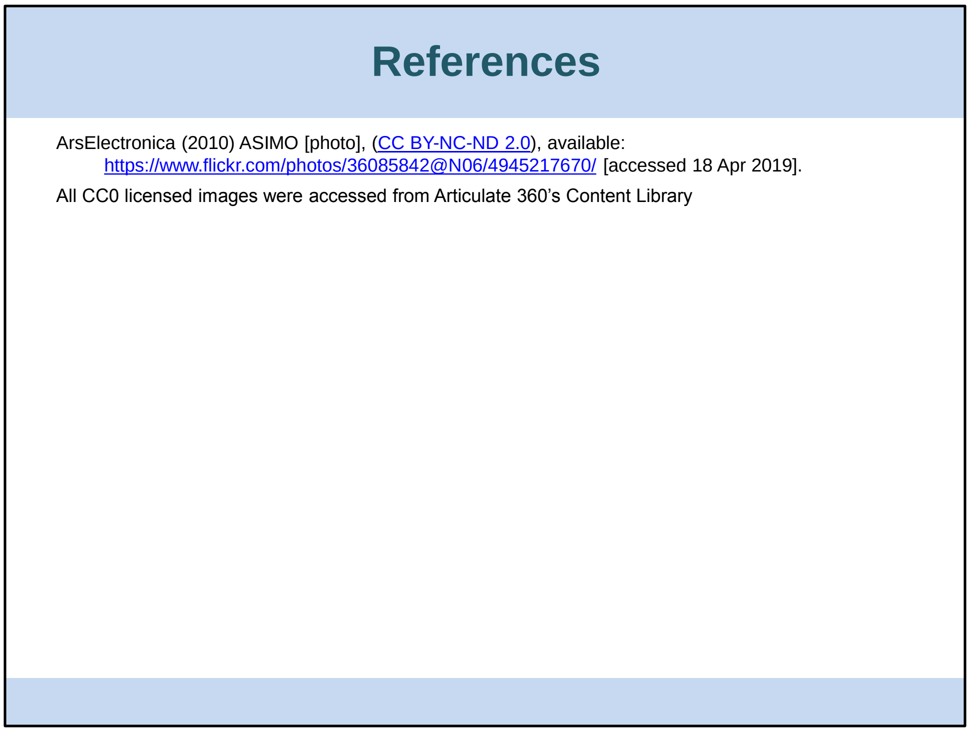
Properties

On passing, 'Finish' button:	Goes to next slide
On failing, 'Finish' button:	Goes to next slide
Allow user to leave quiz:	At any time
User may view slides after quiz:	Any time
Show quiz in menu as:	Multiple Items



Summary

Introduced Ontologies	Introduced the concept of an ontology and demonstrated its use
Described Categories	Described the concept of categories and how they are used to represent knowledge
Examined Relations and Measures	Examined how to establish relations between categories and how to measure objects



References

ArsElectronica (2010) ASIMO [photo], (CC BY-NC-ND 2.0), available: https://www.flickr.com/photos/36085842@N06/4945217670/ [accessed 18 Apr 2019].

All CC0 licensed images were accessed from Articulate 360's Content Library