Intelligenza Artificiale

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Knowledge representation and reasoning

Ontologies and knowledge base representation

Ivan Heibi
Dipartimento di Filologia Classica e Italianistica (FICLIT)
Ivan.heibi2@unibo.it

Ontologies

What content to put into an agent's knowledge base and how to represent facts about the world?

Complex domains (e.g., shopping on the Internet) require more general and flexible representations.

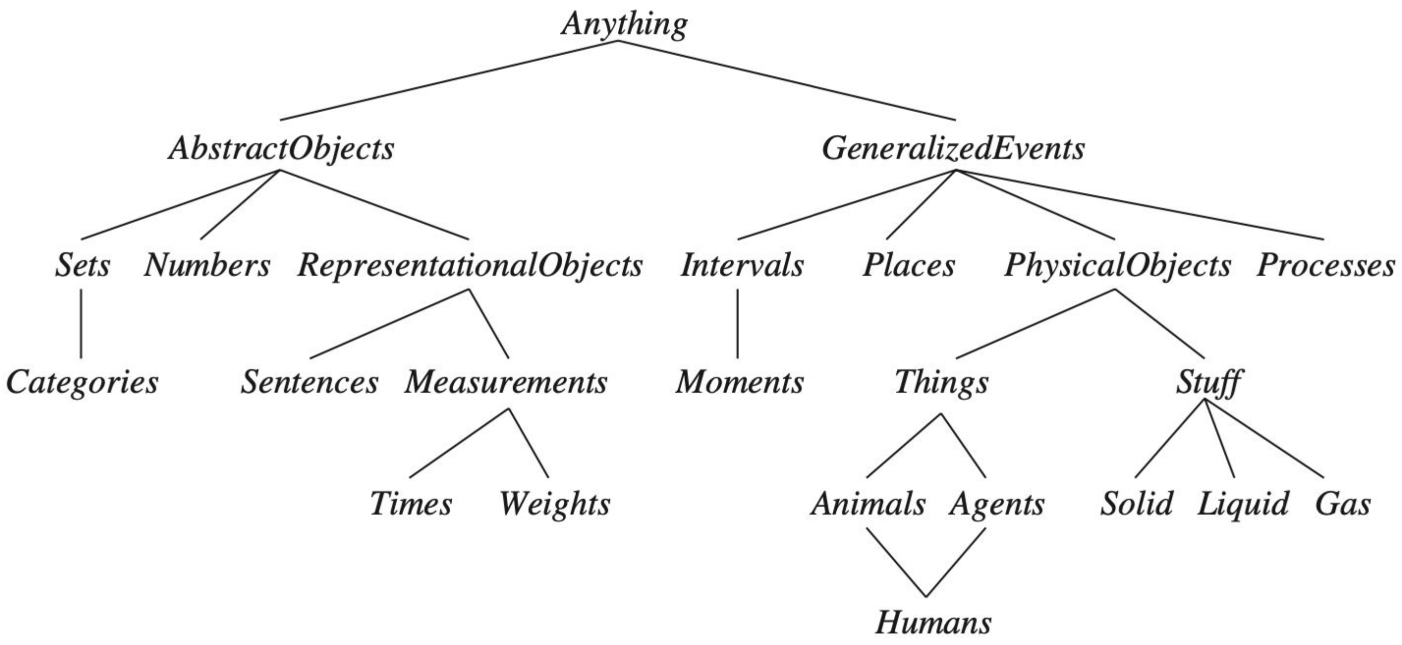
> **>** We need to create representations, concentrating on general concepts: Events, Time, Physical Objects, and Beliefs that occur in many different domains.

An **ontology** aims at organising everything in the world into a hierarchy of categories. More precisely, "An ontology is explicit specification of a conceptualization".

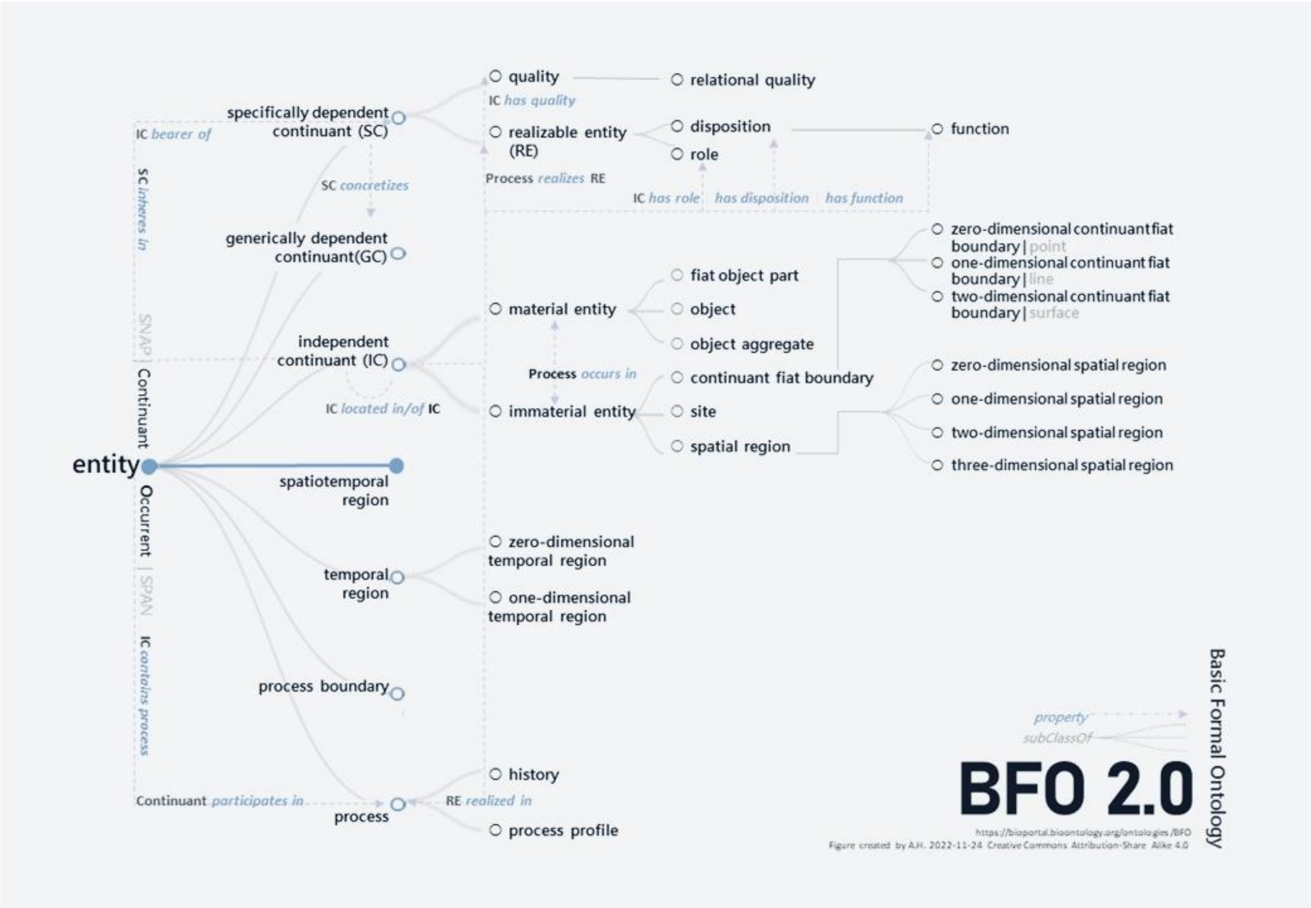
Representing abstract concepts is called **ontological engineering**.

Upper ontology of the world

The **upper ontology** is a general framework of concepts, it aims to capture universal and abstract concepts that are common to a wide range of knowledge areas



Upper ontology – BFO



https://www.wikidata.org/wiki/Q4866972#/media/File:BFO_ontology.jpg

Upper ontology characteristics

Standardization, a common set of terms and relationships that can be adopted uniformly across different applications and domains

Interoperability, enable the exchange of information between diverse sources.

Reusability, concepts can be reused in various domains, (reducing the need to reinvent the wheel for common, foundational concepts).

Categories and objects

Although interaction with the world takes place at the level of individual objects, much reasoning takes place at the level of categories.

For example, we interact with a cat named Lilly as an **individual object**, but when we want to buy food we deal with her as a **category.**

In FOL we can present “basketball” in two ways:

- **as predicate:** *Basketball(b)*
- **as an object:** *Member(b ,Basketballs)* and *Subset(Basketballs, Balls)*

Inheritance

A class of objects (category) ***inherit*** *the properties and behaviors of its* ***superclass***

For example:

- all instances of the category *Food* are edible;
- *Fruit* is a subclass of *Food* and *Apples* is a subclass of *Fruit*,
- **we infer that every apple is edible** (the individual apples inherit the property of edibility, following their membership in the *Food* category).

Objects and categories In FOL

An object is a member of a category.

$BB9 \in Basketballs$

A category is a subclass of another category.

$Basketballs \subset Balls$

All members of a category have some properties.

$(x \in Basketballs) \Rightarrow Spherical(x)$

Members of a category can be recognized by some properties.

$Orange(x) \wedge Round(x) \wedge Diameter(x) = 9.5 \wedge x \in Balls \Rightarrow x \in Basketballs$

A category as a whole has some properties.

$Dogs \in DomesticatedSpecies$

Sets

Sets

A set is a well-defined collection of objects (namely, elements of the set)

Formally, a set can be defined as: {1,4,8,3,6}

If x is an element of set A then: $x \in A$

If x is <u>not an</u> element of set A then: $x \notin A$

Types of sets

If every element of a set B is also an element of another set A, then we say B is a **<u>subset</u>** of set A

Given: A = {B,I,G,S,E,T} and B={E,T}

Then B is a subset of A: $B \subseteq A$

Two sets A and B are **<u>equal</u>** (A = B) if they have the same type and number of elements

$B \subseteq A$ and $A \subseteq B$

Disjoint sets

Two sets A and B are **disjoint**, if they do not have any element in common

Different elements:
A: {1, 2, 3}
B: {4, 5, 6}

Type of elements:
A: {A, B, C}
B: {1, 2, 3, 4}

Disjoint subsets:
C: {1, 2, 3, 4, 5, 6, 7, 8, 9}

- A: {1, 3, 5, 7, 9}
- B: {2, 4, 6, 8}

Operations on sets: union and intersection

The **union** combines two sets into a new set containing all unique elements from both sets. $A \cup B$

$\{1, 2, 3, 4\} \cup \{2, 5\} = \{1, 2, 3, 4, 5\}$

The **intersection** between two sets includes only elements that are present in both sets. $A \cap B$

$\{1, 2, 3, 4\} \cap \{2, 5\} = \{2\}$

Example

Five friends are planning a movie night and want to find a film everyone enjoys. Given everyone's preferred genres (which are: comedy, action, drama, adventure, and horror), use sets to determine the best options!

Example

Five friends are planning a movie night and want to find a film everyone enjoys. Given everyone's preferred genres (which could be: comedy, action, drama, adventure, and horror), use sets to determine the best options!

F1 = { <genres> }
F2 = { <genres> }
F3 = { <genres> }
F4 = { <genres> }
F5 = { <genres> }

Example

Five friends are planning a movie night and want to find a film everyone enjoys. Given everyone's preferred genres (which could be: comedy, action, drama, adventure, and horror), use sets to determine the best options!

F1 = { <genres> }
F2 = { <genres> }
F3 = { <genres> }
F4 = { <genres> }
F5 = { <genres> }

$F1 \cap F2 \cap F3 \cap F4 \cap F5$

Example

Five friends are planning a movie night and want to find a film everyone enjoys. Given everyone's preferred genres (which could be: comedy, action, drama, adventure, and horror), use sets to determine the best options!

F1 = { “comedy”, “action”, “horror” }
F2 = { “horror” }
F3 = { “comedy”, “horror” }
F4 = { “drama”, “horror”, “action” }
F5 = { “horror”, “action”, “adventure” }

F1 ∩ F2 ∩ F3 ∩ F4 ∩ F5 = {“horror”}

Example

Five friends are planning a movie night and want to find a film everyone enjoys. Given everyone's preferred genres (which could be: comedy, action, drama, adventure, and horror), use sets to determine the best options!

F1 = { "comedy", "action", "horror" }
F2 = { "comedy", "drama" }
F3 = { "comedy", "horror" }
F4 = { "drama", "horror", "action" }
F5 = { "horror", "action", "adventure" }

What if the intersection is empty?

Example

Five friends are planning a movie night and want to find a film everyone enjoys. Given everyone's preferred genres (which could be: comedy, action, drama, adventure, and horror), use sets to determine the best options!

F1 = { "comedy", "action", "horror" }
F2 = { "comedy", "drama" }
F3 = { "comedy", "horror" }
F4 = { "drama", "horror", "action" }
F5 = { "horror", "action", "adventure" }

1. Define a majority value = 3
2. calculate all possible intersections using 4 sets
3. if one intersection have at least 1 element stop and return that set, otherwise do again step (2) using 3 sets

Example

Five friends are planning a movie night and want to find a film everyone enjoys. Given everyone's preferred genres (which could be: comedy, action, drama, adventure, and horror), use sets to determine the best options!

F1 = { “comedy”, “action”, “horror” }
F2 = { “comedy”, “drama” }
F3 = { “comedy”, “horror” }
F4 = { “drama”, “horror”, “action” }
F5 = { “horror”, “action”, “adventure” }

At least 4 people like:
{“horror”}

Example

Five friends are planning a movie night and want to find a film everyone enjoys. Given everyone's preferred genres (which could be: comedy, action, drama, adventure, and horror), use sets to determine the best options!

F1 = { "comedy", "action", "horror" }
F2 = { "comedy", "drama" }
F3 = { "comedy", "horror" }
F4 = { "drama", "horror", "action" }
F5 = { "horror", "action", "adventure" }

1. majority = 3
2. Count how many like each genre
3. select the genre liked the most

Relations between categories

We want to state relations between categories that are not subclasses of each other.

For example:

➔ if we just say that *Undergraduates* and *GraduateStudents* are subclasses of *Student*, then we have not said that an undergraduate cannot also be a graduate student.

Two or more categories are:

- **disjoint** if they have no members in common
- form an **exhaustive decomposition** of another category C (e.g., university students)– if each object of C belongs to at least one of these categories
- form a **partition** if all categories in the **exhaustive decomposition** are disjoint

Relations between categories – example

Disjoint({Animals, Trees})

ExhaustiveDecomposition({Action, Comedy, Drama, Romance}, ***Movies*** *)*

Partition({Africa, Antarctica, Asia, Europe, North America, South America, and Oceania}, ***Continents*** *)*

Physical composition

The idea that one object can be part of another is a familiar one.

For example, a nose is part of the head, Italy is part of Europe, and a chapter is part of a book

The ***PartOf*** relation is used to say that one thing is part of another. Objects can be grouped into *PartOf* hierarchies, (to indicate the Subset hierarchy)

For example:

PartOf (Bologna, Italy)
PartOf (Italy, Europe)
PartOf (Europe, Earth)

Reasoning Systems for Categories

Family Systems specially designed for organizing and reasoning with categories, are:

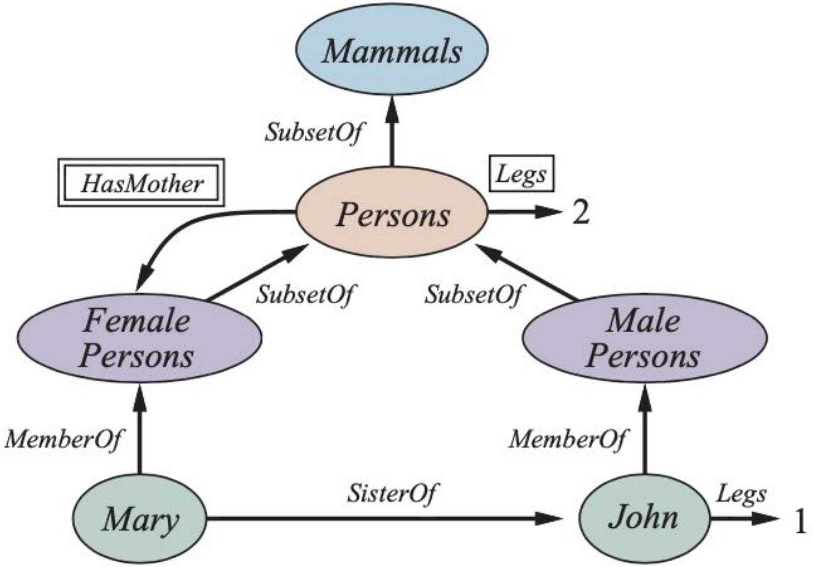
- **Semantic networks** providing a graphical aids for visualizing a knowledge base
- **Description logics** providing a formal language for constructing and combining category definitions and efficient algorithms for deciding subset and superset relationships between categories.

Semantic networks

A typical graphical notation displays object or category names in ovals or boxes, and connects them with labeled links

The single-boxed link is used to assert properties of every member of a category.

The double-boxed link is used to assert a relation between **an object person and his mother (<u>not the entire category</u>)**.



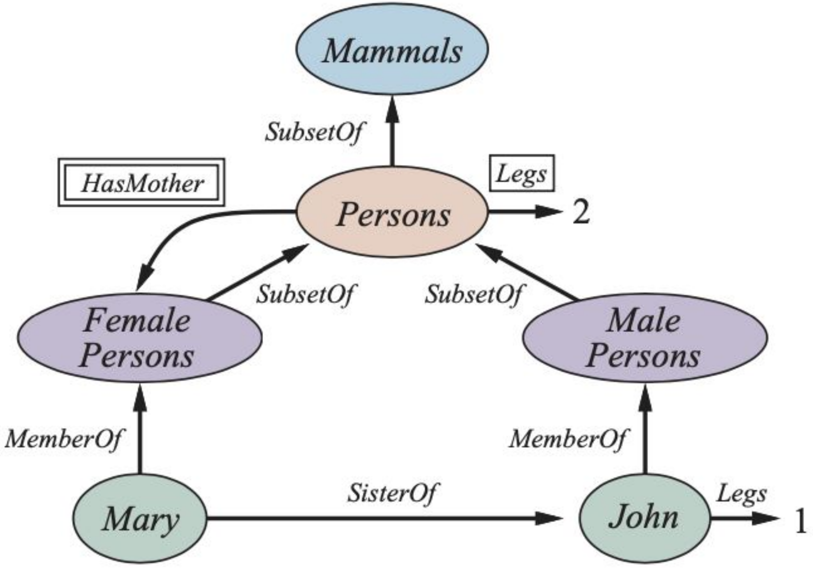
Semantic networks

A typical graphical notation displays object or category names in ovals or boxes, and connects them with labeled links

The double-boxed link is used to assert a relation between **an object person and his mother (<u>not the entire category</u>)**.

Can we draw a *HasMother* link (with no box) from Persons to *FemalePersons*?

Answer is: NO



Semantic networks – multiple inheritance

What if a category is a subset of more than one other category; what does it inherit?

This is called **multiple inheritance**.
> In such cases, the inheritance algorithm might find two or more conflicting values.

For this reason, multiple inheritance is banned in some object-oriented programming languages, e.g. JAVA

For example,
> if *Bat* is a subclass of both *Birds* and *Mammals* and both have the property "*make sound*", which one should he inherit?

Self assessment

https://forms.gle/ZgYr4GjXnGT8auVu7

Description logics

Description logics are notations that are designed to make it easier to describe definitions and properties of categories.

Description logic systems evolved from semantic networks in response to <u>pressure to formalize what the networks mean</u> while retaining the emphasis on taxonomic structure as an organizing principle.

Inference tasks for description logics are mainly:

- **subsumption** (checking if one category is a subset of another by comparing their definitions)
- **classification** (checking whether an object belongs to a category).
- **consistency of a category definition**—whether the membership criteria are logically satisfiable.

Description logics: subsumption

Subsumption checks whether a category (subclass) is a subset of another (superclass)

➔ All objects belonging to the subclass also belong to the superclass.

For example:

$$\text{Dog} \equiv \text{Pet} \sqcap \text{HasFur}$$

$$\text{Pet} \equiv \text{Mammal} \sqcap \text{NonHuman}$$

Subsumption is used to check if Dog is a subset of Pet, and it compares the definitions of both categories.

➔ In this case, all dogs are pets with fur, and all pets are non-human mammals. Therefore, Dog is a subset of Pet.

Description logics: classification

Classification checks whether an object (instance) belongs to a category (class).

➔ Verifies whether the <u>properties</u> of the instance satisfy the membership criteria of the class.

For example:
Let's consider the instance Fido with the properties dog and long fur.
With the classification we check if Fido belongs to the Dog class.

➔ Verifying whether Fido's properties satisfy the definition of Dog (pet with fur). In this case, Fido is classified as a Dog.

Description logics: consistency

Consistency checks if the definition of a category is logically satisfiable.

➔ Verifies if there exist objects that can simultaneously satisfy all the membership criteria of the category.

For example:

Let's consider the following definition:

Bird ≡ Animal ⊓ HasWings ⊓ CanFly

The definition is inconsistent because it implies that all birds must have wings and the ability to fly. However, there are flightless birds (e.g., ostriches, penguins) that do not possess the ability to fly.

Description logics – example

CLASSIC is a description logic of objects in terms of their relations to other known objects, and their level of intensional structure.

$$
\begin{array}{rcl}
\textit{Concept} & \rightarrow & \textbf{Thing} \mid \textit{ConceptName} \\
& \mid & \textbf{And}(\textit{Concept}, \ldots) \\
& \mid & \textbf{All}(\textit{RoleName}, \textit{Concept}) \\
& \mid & \textbf{AtLeast}(\textit{Integer}, \textit{RoleName}) \\
& \mid & \textbf{AtMost}(\textit{Integer}, \textit{RoleName}) \\
& \mid & \textbf{Fills}(\textit{RoleName}, \textit{IndividualName}, \ldots) \\
& \mid & \textbf{SameAs}(\textit{Path}, \textit{Path}) \\
& \mid & \textbf{OneOf}(\textit{IndividualName}, \ldots) \\
\textit{Path} & \rightarrow & [\textit{RoleName}, \ldots] \\
\textit{ConceptName} & \rightarrow & \textit{Adult} \mid \textit{Female} \mid \textit{Male} \mid \ldots \\
\textit{RoleName} & \rightarrow & \textit{Spouse} \mid \textit{Daughter} \mid \textit{Son} \mid \ldots
\end{array}
$$

Description logics – example

CLASSIC is a description logic of objects in terms of their relations to other known objects, and their level of intensional structure.

$$
\begin{array}{rcl}
\mathit{Concept} & \rightarrow & \textbf{Thing} \mid \mathit{ConceptName} \\
& \mid & \textbf{And}(\mathit{Concept}, \ldots) \\
& \mid & \textbf{All}(\mathit{RoleName}, \mathit{Concept}) \\
& \mid & \textbf{AtLeast}(\mathit{Integer}, \mathit{RoleName}) \\
& \mid & \textbf{AtMost}(\mathit{Integer}, \mathit{RoleName}) \\
& \mid & \textbf{Fills}(\mathit{RoleName}, \mathit{IndividualName}, \ldots) \\
& \mid & \textbf{SameAs}(\mathit{Path}, \mathit{Path}) \\
& \mid & \textbf{OneOf}(\mathit{IndividualName}, \ldots) \\
\mathit{Path} & \rightarrow & [\mathit{RoleName}, \ldots] \\
\mathit{ConceptName} & \rightarrow & \mathit{Adult} \mid \mathit{Female} \mid \mathit{Male} \mid \ldots \\
\mathit{RoleName} & \rightarrow & \mathit{Spouse} \mid \mathit{Daughter} \mid \mathit{Son} \mid \ldots
\end{array}
$$

Bachelors are unmarried adult males:

$$Bachelor = And(Unmarried, Adult, Male)$$

In FOL:

$$Bachelor(x) \Leftrightarrow Unmarried(x) \wedge Adult(x) \wedge Male(x)$$

Exercises

Exercise

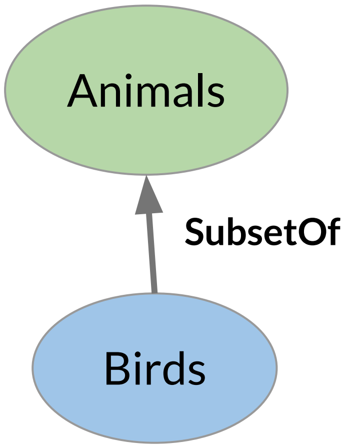
Represent the following sentences into a semantic network.

- Birds are animals.
- Birds can fly and lay eggs.
- Albatros is a bird.
- Donald is a bird.
- Tracy is an albatros.

Exercise

Represent the following sentences into a semantic network.

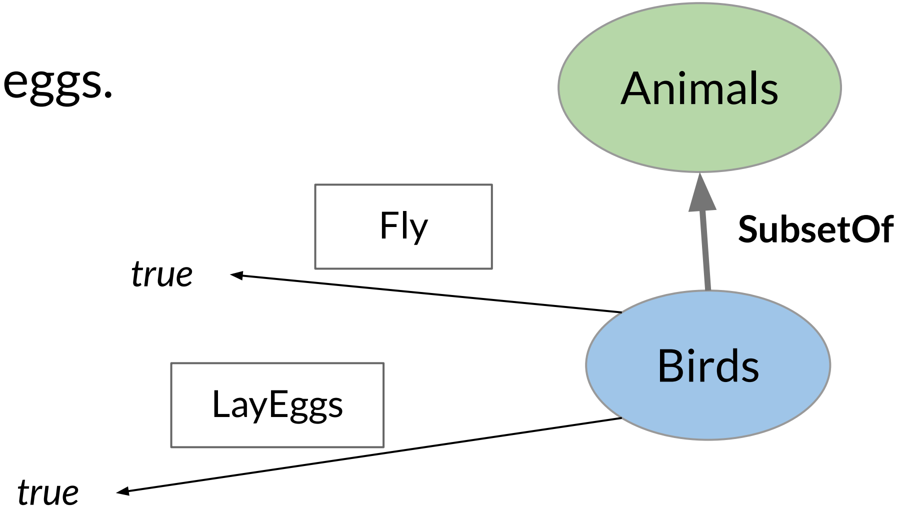
- Birds are animals.
- Birds can fly and lay eggs.
- Albatros is a bird.
- Donald is a bird.
- Tracy is an albatros.



Exercise

Represent the following sentences into a semantic network.

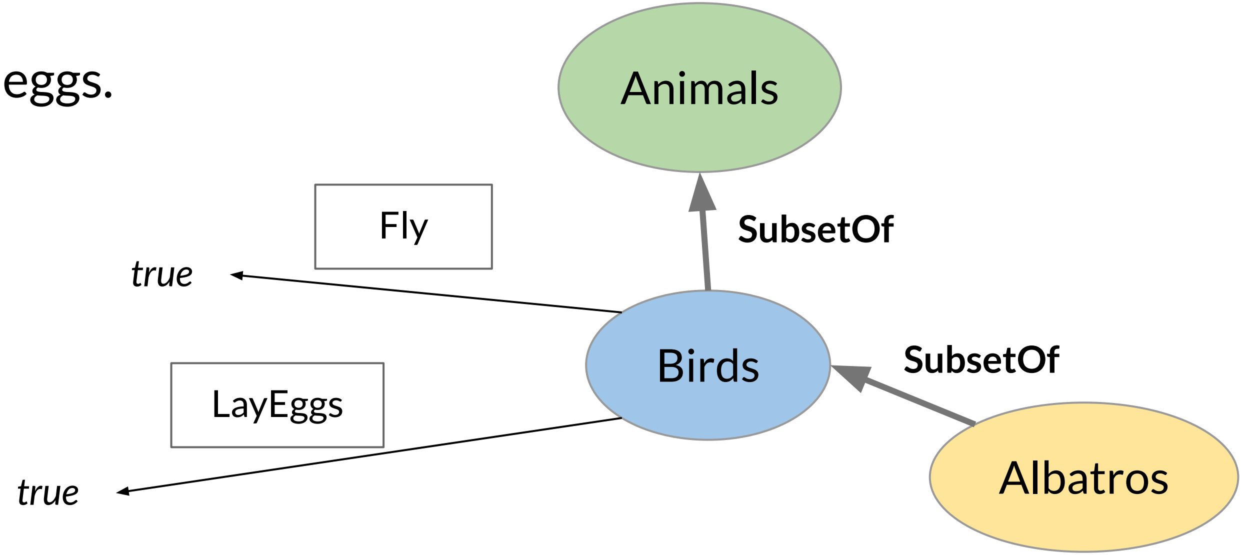
- Birds are animals.
- Birds can fly and lay eggs.
- Albatros is a bird.
- Donald is a bird.
- Tracy is an albatros.



Exercise

Represent the following sentences into a semantic network.

- Birds are animals.
- Birds can fly and lay eggs.
- Albatros is a bird.
- Donald is a bird.
- Tracy is an albatros.



Exercise

Represent the following sentences into a semantic network.

- Birds are animals.
- Birds can fly and lay eggs.
- Albatros is a bird.
- Donald is a bird.
- Tracy is an albatros.