Fundamentals of Artificial Intelligence and Knowledge Representation (Module 2)

Last update: 24 November 2023

Academic Year 2023 – 2024

Alma Mater Studiorum · University of Bologna

Contents

1 Propositional and first order logic

See `Languages and Algorithms for AI (module 2)`.

2 Ontologies

Ontology Formal (non-ambiguous) and explicit (obtainable through a finite sound procedure) description of a domain. Ontology

Category Can be organized hierarchically on different levels of generality. Category

Object Belongs to one or more categories. Object

Upper/general ontology Ontology focused on the most general domain. Upper/general ontology

Properties:

- Should be applicable to almost any special domain.
- Combining general concepts should not incur in inconsistences.

Approaches to create ontologies:

- Created by philosophers/logicians/researchers.
- Automatic knowledge extraction from well-structured databases.
- Created from text documents (e.g. web).
- Crowd-sharing information.

2.1 Categories

Category Used in human reasoning when the goal is category-driven (in contrast to specific-instance-driven). Category

In first order logic, categories can be represented through:

Predicate A predicate to tell if an object belongs to a category (e.g. `Car(c1)` indicates that `c1` is a car). Predicate categories

Reification Represent categories as objects as well (e.g. $\texttt{c1} \in \texttt{Car}$). Reification

2.1.1 Reification properties and operations

Membership Indicates if an object belongs to a category. (e.g. $\texttt{c1} \in \texttt{Car}$). Membership

Subclass Indicates if a category is a subcategory of another one. (e.g. $\texttt{Car} \subset \texttt{Vehicle}$). Subclass

Necessity Members of a category enjoy some properties (e.g. $(\texttt{x} \in \texttt{Car}) \rightarrow \texttt{hasWheels(x)}$). Necessity

Sufficiency Sufficient conditions to be part of a category (e.g. $\texttt{hasPlate(x)} \land \texttt{hasWheels(x)} \rightarrow \texttt{x} \in \texttt{Car}$). Sufficiency

Category-level properties Category themselves can enjoy properties (e.g. $\texttt{Car} \in \texttt{VehicleType}$) Category-level properties

Disjointness Given a set of categories S, the categories in S are disjoint iff they all have different objects: Disjointness

$$\texttt{disjoint}(S) \iff (\forall c_1, c_2 \in S, c_1 \neq c_2 \rightarrow c_1 \cap c_2 = \emptyset)$$

Exhaustive decomposition Given a category c and a set of categories S, S is an exhaustive decomposition of c iff any element in c belongs to at least a category in S: Exhaustive decomposition

$$\texttt{exhaustiveDecomposition}(S,\ c) \iff (\forall o \in c \iff \exists c_2 \in S : o \in c_2)$$

Partition Given a category c and a set of categories S, S is a partition of c when: Partition

$$\texttt{partition}(S,\ c) \iff \texttt{disjoint}(S) \wedge \texttt{exhaustiveDecomposition}(S,\ c)$$

2.1.2 Physical composition

Objects (meronyms) are part of a whole (holonym).

Part-of If the objects have a structural relation (e.g. `partOf(cylinder1, engine1)`). Part-of

Properties:

Transitivity `partOf(x, y)` $\wedge$ `partOf(y, z)` $\rightarrow$ `partOf(x, z)`

Reflexivity `partOf(x, x)`

Bunch-of If the objects do not have a structural relation. Useful to define a composition of countable objects (e.g. `bunchOf(nail1, nail3, nail4)`). Bunch-of

2.1.3 Measures

A property of objects.

Quantitative measure Something that can be measured using a unit (e.g. `length(table1)` = `cm(80)`). Quantitative measure

Qualitative measures propagate when using `partOf` or `bunchOf` (e.g. the weight of a car is the sum of its parts).

Qualitative measure Something that can be measured using terms with a partial or total order relation (e.g. {`good`, `neutral`, `bad`}). Qualitative measure

Qualitative measures do not propagate when using `partOf` or `bunchOf`.

Fuzzy logic Provides a semantics to qualitative measures (i.e. convert qualitative to quantitative). Fuzzy logic

2.1.4 Things vs stuff

Intrinsic property Related to the substance of the object. It is retained when the object is divided (e.g. water boils at 100°C). Intrinsic property

Extrinsic property Related to the structure of the object. It is not retained when the object is divided (e.g. the weight of an object changes when split). Extrinsic property

Substance Category of objects with only intrinsic properties. Substance

Stuff The most general substance category. Stuff

Count noun Category of objects with only extrinsic properties. Count noun

Things The most general object category. Things

2.2 Semantic networks

Graphical representation of objects and categories connected through labelled links. Semantic networks

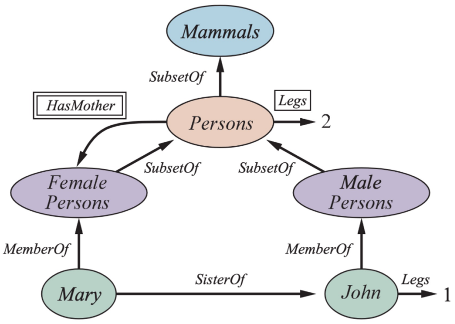

Figure 2.1: Example of semantic network

Objects and categories Represented using the same symbol.

Links Four different types of links:

- Relation between objects (e.g. `SisterOf`).
- Property of a category (e.g. 2 `Legs`).
- Is-a relation (e.g. `SubsetOf`).
- Property of the members of a category (e.g. `HasMother`).

Single inheritance reasoning Starting from an object, check if it has the queried property. If not, iteratively move up to the category it belongs to and check for the property. Single inheritance reasoning

Multiple inheritance reasoning Reasoning is not possible as it is not clear which parent to choose. Multiple inheritance reasoning

Limitations Compared to first order logic, semantic networks do not have:

- Negations.
- Universally and existentially quantified properties.
- Disjunctions.
- Nested function symbols.

Many semantic network systems allow to attach special procedures to handle special cases that the standard inference algorithm cannot handle. This approach is powerful but does not have a corresponding logical meaning.

Advantages With semantic networks it is easy to attach default properties to categories and override them on the objects (i.e. `Legs` of `John`).

2.3 Frames

Knowledge that describes an object in terms of its properties. Each frame has: Frames

- An unique name
- Properties represented as pairs <slot - filler>

Example.

```
(
    toronto
        <:Instance-Of City>
        <:Province ontario>
        <:Population 4.5M>
)
```

Prototype Members of a category used as comparison metric to determine if another object belongs to the same class (i.e. an object belongs to a category if it is similar enough to the prototypes of that category). Prototype

Defeasible value Value that is allowed to be different when comparing an object to a prototype. Defeasible value

Facets Additional information contained in a slot for its filler (e.g. default value, type, domain). Facets

Procedural information Fillers can be a procedure that can be activated by specific facets:

`if-needed` Looks for the value of the slot.

`if-added` Adds a value.

`if-removed` Removes a value.

Example.

```
(
    toronto
        <:Instance-Of City>
        <:Province ontario>
        <:Population [if-needed QueryDB]>
)
```