

Fundamentals of Artificial Intelligence and Knowledge Representation (Module 2)

Last update: 02 December 2023

Academic Year 2023 – 2024
Alma Mater Studiorum · University of Bologna

Contents

1	Propositional and first order logic	1
2	Ontologies	2
2.1	Categories	2
2.1.1	Reification properties and operations	2
2.1.2	Physical composition	3
2.1.3	Measures	3
2.1.4	Things vs stuff	3
2.2	Semantic networks	4
2.3	Frames	4
3	Description logic	6
3.1	Syntax	6
3.2	Semantics	7
3.2.1	Concept-forming operators	7
3.2.2	Sentences	7
3.2.3	Interpretation	7
3.3	Reasoning	8
3.3.1	T-box reasoning	8
3.3.2	A-box reasoning	8
3.3.3	Computing subsumptions	9
3.3.4	Open world assumption	9
3.4	Expanding description logic	9
3.5	Description logics family	10
4	Web reasoning	11
4.1	Semantic web	11
4.2	Knowledge graphs	12

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:		
<ul style="list-style-type: none">• Should be applicable to almost any special domain.• Combining general concepts should not incur in inconsistencies.		
Approaches to create ontologies:		
<ul style="list-style-type: none">• 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. <code>Car(c1)</code> indicates that <code>c1</code> is a car).	Predicate categories
Reification	Represent categories as objects as well (e.g. <code>c1 ∈ Car</code>).	Reification

2.1.1 Reification properties and operations

Membership	Indicates if an object belongs to a category. (e.g. <code>c1 ∈ Car</code>).	Membership
Subclass	Indicates if a category is a subcategory of another one. (e.g. <code>Car ⊂ Vehicle</code>).	Subclass
Necessity	Members of a category enjoy some properties (e.g. $(x \in \text{Car}) \rightarrow \text{hasWheels}(x)$).	Necessity
Sufficiency	Sufficient conditions to be part of a category (e.g. $\text{hasPlate}(x) \wedge \text{hasWheels}(x) \rightarrow x \in \text{Car}$).	Sufficiency
Category-level properties	Category themselves can enjoy properties (e.g. <code>Car ∈ VehicleType</code>)	Category-level properties

Disjointness	Given a set of categories S , the categories in S are disjoint iff they all have different objects:	Disjointness
	$\text{disjoint}(S) \iff (\forall c_1, c_2 \in S, c_1 \neq c_2 \rightarrow c_1 \cap c_2 = \emptyset)$	
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
	$\text{exhaustiveDecomposition}(S, c) \iff (\forall o \in c \iff \exists c_2 \in S : o \in c_2)$	
Partition	Given a category c and a set of categories S , S is a partition of c when:	Partition
	$\text{partition}(S, c) \iff \text{disjoint}(S) \wedge \text{exhaustiveDecomposition}(S, c)$	
2.1.2 Physical composition		
	Objects (meronyms) are part of a whole (holonym).	
Part-of	If the objects have a structural relation (e.g. <code>partOf(cylinder1, engine1)</code>). Properties:	Part-of
	Transitivity $\text{partOf}(x, y) \wedge \text{partOf}(y, z) \rightarrow \text{partOf}(x, z)$ Reflexivity $\text{partOf}(x, x)$	
Bunch-of	If the objects do not have a structural relation. Useful to define a composition of countable objects (e.g. <code>bunchOf(nail1, nail3, nail4)</code>).	Bunch-of
2.1.3 Measures		
	A property of objects.	
Quantitative measure	Something that can be measured using a unit (e.g. <code>length(table1) = cm(80)</code>). Qualitative measures propagate when using <code>partOf</code> or <code>bunchOf</code> (e.g. the weight of a car is the sum of its parts).	Quantitative measure
Qualitative measure	Something that can be measured using terms with a partial or total order relation (e.g. <code>{good, neutral, bad}</code>). Qualitative measures do not propagate when using <code>partOf</code> or <code>bunchOf</code> .	Qualitative measure
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

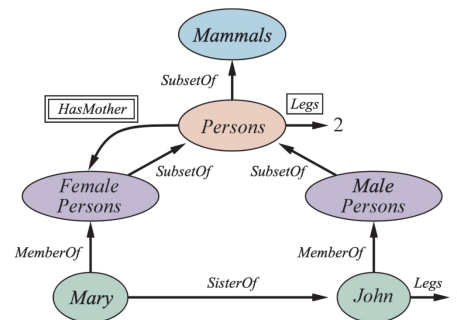


Figure 2.1: Example of semantic network

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]>  
)
```

3 Description logic

3.1 Syntax

Logical symbols Symbols with fixed meaning.

Logical symbols

Punctuation () []

Positive integers

Concept-forming operators ALL, EXISTS, FILLS, AND

Connectives \sqsubseteq , \doteq , \rightarrow

Non-logical symbols Domain-dependant symbols.

Non-logical symbols

Atomic concepts Categories (CamelCase, e.g. Person).

Roles Used to describe objects (:CamelCase, e.g. :Height).

Constants (camelCase, e.g. johnDoe).

Complex concept Concept-forming operators can be used to combine atomic concepts and form complex concepts. A well-formed concept follows the conditions:

Complex concept

- An atomic concept is a concept.
- If r is a role and d is a concept, then $[ALL\ r\ d]$ is a concept.
- If r is a role and n is a positive integer, then $[EXISTS\ n\ r]$ is a concept.
- If r is a role and c is a constant, then $[FILLS\ r\ c]$ is a concept.
- If $d_1 \dots d_n$ are concepts, then $[AND\ d_1 \dots d_n]$ is a concept.

Sentence Connectives can be used to combine concepts and form sentences. A well-formed sentence follows the conditions:

Sentence

- If d_1 and d_2 are concepts, then $(d_1 \sqsubseteq d_2)$ is a sentence.
- If d_1 and d_2 are concepts, then $(d_1 \doteq d_2)$ is a sentence.
- If c is a constant and d is a concept, then $(c \rightarrow d)$ is a sentence.

Knowledge base Collection of sentences.

Knowledge base

Constants are individuals of the domain.

Concepts are categories of individuals.

Roles are binary relations between individuals.

Assertion box (A-box) List of facts about individuals.

Assertion box
(A-box)

Terminological box (T-box) List of sentences (axioms) about concepts.

Terminological box
(T-box)

3.2 Semantics

3.2.1 Concept-forming operators

Let r be a role, d be a concept, c be a constant and n a positive integer. The semantics of concept-forming operators are:

Concept-forming operators

$[ALL \ r \ d]$ Individuals r -related to the individuals of the category d .

Example. $[ALL \ :HasChild \ Male]$ individuals that have zero children or only male children.

$[EXISTS \ n \ r]$ Individuals r -related to at least n other individuals.

Example. $[EXISTS \ 1 \ :Child]$ individuals with at least one child.

$[FILLS \ r \ c]$ Individuals r -related to the individual c .

Example. $[FILLS \ :Child \ john]$ individuals with child john.

$[AND \ d_1 \dots d_n]$ Individuals belonging to all the categories $d_1 \dots d_n$.

3.2.2 Sentences

Sentences are expressions with truth values in the domain. Let d be a concept and c be a constant. The semantics of sentences are:

Sentences

$d_1 \sqsubseteq d_2$ Concept d_1 is subsumed by d_2 .

Example. $PhDStudent \sqsubseteq Student$ as every PhD is also a student.

$d_1 \doteq d_2$ Concept d_1 is equivalent to d_2 .

Example. $PhDStudent \doteq [AND \ Student \ :Graduated \ :HasFunding]$

$c \rightarrow d$ The individual c satisfies the description of the concept d .

Example. $federico \rightarrow Professor$

3.2.3 Interpretation

Interpretation An interpretation \mathcal{I} in description logic is a pair $(\mathcal{D}, \mathcal{I})$ where:

Interpretation

- \mathcal{D} is the domain.
- \mathcal{I} is the interpretation mapping.

Constant Let c be a constant, $\mathcal{I}[c] \in \mathcal{D}$.

Atomic concept Let a be an atomic concept, $\mathcal{I}[a] \subseteq \mathcal{D}$.

Role Let r be a role, $\mathcal{I}[r] \subseteq \mathcal{D} \times \mathcal{D}$.

Thing The concept **Thing** corresponds to the domain: $\mathcal{I}[\text{Thing}] = \mathcal{D}$.

$[ALL \ r \ d]$

$$\mathcal{I}[[ALL \ r \ d]] = \{x \in \mathcal{D} \mid \forall y : \langle x, y \rangle \in \mathcal{I}[r] \text{ then } y \in \mathcal{I}[d]\}$$

$[EXISTS \ n \ r]$

$$\mathcal{I}[[EXISTS \ n \ r]] = \{x \in \mathcal{D} \mid \text{exists at least } n \text{ distinct } y : \langle x, y \rangle \in \mathcal{I}[r]\}$$

[FILLS $r \ c$]

$$\mathcal{I}[\text{[FILLS } r \ c]] = \{x \in \mathcal{D} \mid \langle x, \mathcal{I}[c] \rangle \in \mathcal{I}[r]\}$$

[AND $d_1 \dots d_n$]

$$\mathcal{I}[\text{[AND } d_1 \dots d_n]] = \mathcal{I}[d_1] \cap \dots \cap \mathcal{I}[d_n]$$

Model Given an interpretation $\mathcal{J} = (\mathcal{D}, \mathcal{I})$, a sentence is true under \mathcal{J} ($\mathcal{J} \models \text{sentence}$) if:

Model

- $\mathcal{J} \models (c \rightarrow d)$ iff $\mathcal{I}[c] \in \mathcal{I}[d]$.
- $\mathcal{J} \models (d_1 \sqsubseteq d_2)$ iff $\mathcal{I}[d_1] \subseteq \mathcal{I}[d_2]$.
- $\mathcal{J} \models (d_1 \doteq d_2)$ iff $\mathcal{I}[d_1] = \mathcal{I}[d_2]$.

Given a set of sentences S , \mathcal{J} models S if $\mathcal{J} \models S$.

Entailment A set of sentences S logically entails a sentence α if:

Entailment

$$\forall \mathcal{J} : (\mathcal{J} \models S) \rightarrow (\mathcal{J} \models \alpha)$$

3.3 Reasoning

3.3.1 T-box reasoning

Given a knowledge base of a set of sentences S , we would like to be able to determine the following:

Satisfiability A concept d is satisfiable w.r.t. S if:

Satisfiability

$$\exists \mathcal{J}, (\mathcal{J} \models S) : \mathcal{J}[d] \neq \emptyset$$

Subsumption A concept d_1 is subsumed by d_2 w.r.t. S if:

Subsumption

$$\forall \mathcal{J}, (\mathcal{J} \models S) : \mathcal{J}[d_1] \subseteq \mathcal{J}[d_2]$$

Equivalence A concept d_1 is equivalent to d_2 w.r.t. S if:

Equivalence

$$\forall \mathcal{J}, (\mathcal{J} \models S) : \mathcal{J}[d_1] = \mathcal{J}[d_2]$$

Disjointness A concept d_1 is disjoint to d_2 w.r.t. S if:

Disjointness

$$\forall \mathcal{J}, (\mathcal{J} \models S) : \mathcal{J}[d_1] \cap \mathcal{J}[d_2] = \emptyset$$

Theorem 3.3.1 (Reduction to subsumption). Given the concepts d_1 and d_2 , it holds that:

Reduction to subsumption

- d_1 is unsatisfiable $\iff d_1 \sqsubseteq \perp$.
- $d_1 \doteq d_2 \iff d_1 \sqsubseteq d_2 \wedge d_2 \sqsubseteq d_1$.
- d_1 and d_2 are disjoint $\iff (d_1 \sqcap d_2) \sqsubseteq \perp$.

3.3.2 A-box reasoning

Given a constant c , a concept d and a set of sentences S , we can determine the following:

Satisfiability A constant c satisfies the concept d if:

Satisfiability

$$S \models (c \rightarrow d)$$

Note that it can be reduced to subsumption.

3.3.3 Computing subsumptions

Given a knowledge base KB and two concepts d and e , we want to prove:

$$KB \models (d \sqsubseteq e)$$

The following algorithms can be employed:

Structural matching

Structural matching

1. Normalize d and e into a conjunctive form:

$$d = [\text{AND } d_1 \dots d_n] \quad e = [\text{AND } e_1 \dots e_m]$$

2. Check if each part of e is accounted by at least a component of d .

Tableaux-based algorithms

Exploit the following theorem:

Tableaux-based algorithms

$$(KB \models (C \sqsubseteq D)) \iff (KB \cup (x : C \sqcap \neg D)) \text{ is inconsistent}$$

Note: similar to refutation.

3.3.4 Open world assumption

Open world assumption If a sentence cannot be inferred, its truth values is unknown.

Open world assumption

Description logics are based on the open world assumption. To reason in open world assumption, all the possible models are split upon encountering an unknown facts depending on the possible cases (Oedipus example).

3.4 Expanding description logic

It is possible to expand a description logic by:

Adding concept-forming operators Let r be a role, d be a concept, c be a constant and n a positive integer. We can extend our description logic with:

Adding concept-forming operators

[AT-MOST n r] Individuals r -related to at most n other individuals.

Example. [AT-MOST 1 :Child] individuals with only a child.

[ONE-OF $c_1 \dots c_n$] Concept only satisfied by $c_1 \dots c_n$.

Example. Beatles \doteq [ALL :BandMember [ONE-OF john paul george ringo]]

[EXISTS n r d] Individuals r -related to at least n individuals in the category d .

Example. [EXISTS 2 :Child Male] individuals with at least two male children.

Note: this increases the computational complexity of entailment.

Relating roles

Relating roles

[SAME-AS r_1 r_2] Equates fillers of the roles r_1 and r_2

Example. [SAME-AS :CEO :Owner]

Note: this increases the computational complexity of entailment. Role chaining also leads to undecidability.

Adding rules Rules are useful to add conditions (e.g. if d_1 then [FILLS r c]).

Adding rules

3.5 Description logics family

Depending on the number of operators, a description logic can be:

- More expressive.
- Computationally more expensive.
- Undecidable.

Attributive language (\mathcal{AL}) Minimal description logic with:

- Atomic concepts.
- Universal concept (**Thing** or \top).
- Bottom concept (**Nothing** or \perp).
- Atomic negation (only for atomic concepts).
- AND operator (\sqcap).
- ALL operator (\forall).
- [EXISTS 1 r] operator (\exists).

Attributive language complement (\mathcal{ALC}) \mathcal{AL} with negation for concepts.

\mathcal{F}	Functional properties
\mathcal{E}	Full existential quantification
\mathcal{U}	Concept union
\mathcal{C}	Complex concept negation
\mathcal{S}	\mathcal{ALC} with transitive roles
\mathcal{H}	Role hierarchy
\mathcal{R}	Limited complex roles axioms Reflexivity and irreflexivity Roles disjointness
\mathcal{O}	Nominals
\mathcal{I}	Inverse properties
\mathcal{N}	Cardinality restrictions
\mathcal{Q}	Qualified cardinality restrictions
(\mathcal{D})	Datatype properties, data values and data types

Table 3.1: Name and expressivity of logics

4 Web reasoning

4.1 Semantic web

Semantic web Method to represent and reason on the data available on the web. Semantic web aims to preserve the characteristics of the web, this includes:

- Globality.
- Information distribution.
- Information inconsistency of contents and links (as everyone can publish).
- Information incompleteness of contents and links.

Information is structured using ontologies and logic is used as inference mechanism. New knowledge can be derived through proofs.

Uniform resource identifier Naming system to uniquely identify concepts. Each URI correspond to one and only one concept, but multiple URIs can refer to the same concept.

XML Markup language to represent hierarchically structured data. An XML can contain in its preamble the description of the grammar used within the document.

Resource description framework (RDF) XML-based language to represent knowledge. Based on triplets:

`<subject, predicate, object>`
`<resource, attribute, value>`

RDF supports:

Types Using the attribute `type` which can assume an URI as value.

Collections Subjects and objects can be bags, sequences or alternatives.

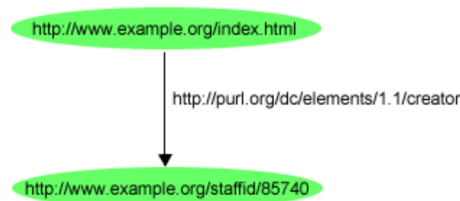
Meta-sentences Reification of the sentences (e.g. "X says that Y...").

RDF schema RDF can be used to describe classes and relations with other classes (e.g. `type`, `subClassOf`, `subPropertyOf`, ...)

Representation

Graph A graph where nodes are subjects or objects and edges are predicates.

Example.



The graph stands for: `http://www.example.org/index.html` has a creator with staff id 85740.

XML

Example.

```
<rdf:RDF
  xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
  xmlns:contact=http://www.w3.org/2000/10/swap/pim/contact#>
  <contact:Person rdf:about="http://www.w3.org/People/EM/
    contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

Database similarities RDF aims to integrate different databases:

- A DB record is a RDF node.
- The name of a column can be seen as a property type.
- The value of a field corresponds to the value of a property.

RDFa Specification to integrate XHTML and RDF.

RDFa

SPARQL Language to query different data sources that support RDF (natively or through a middleware).

SPARQL

Ontology web language (OWL) Ontology based on RDF and description logic fragments. Three level of expressivity are available:

Ontology web language (OWL)

- OWL lite.
- OWL DL.
- OWL full.

An OWL has:

Classes Categories.

Properties Roles and relations.

Instances Individuals.

4.2 Knowledge graphs

Knowledge graph Knowledge graphs overcome the computational complexity of T-box reasoning with semantic web and description logics.

Knowledge graph

- Use a simple vocabulary with a simple but robust corpus of types and properties adopted as a standard.
- Represent a graph with terms as nodes and edges connecting them. Knowledge is therefore represented as triplets (**h**, **r**, **t**) where **h** and **t** are entities and **r** is a relation.
- Logic formulas are removed. T-box and A-box can be seen as the same concept. There is no reasoning but only facts.

- Data does not have a conceptual schema and can come from different sources with different semantics.
- Graph algorithms to traverse the graph and solve queries.

KG quality

Quality

Coverage If the graph has all the required information.

Correctness If the information is correct (can be objective or subjective).

Freshness If the content is up-to-date.

Graph embedding Project entities and relations into a vectorial space for ML applications.

Graph embedding

Entity prediction Given two entities h and t , determine the relation r between them.

Link prediction Given an entity h and a relation t , determine an entity t related to h .