Fundamentals of Artificial Intelligence and Knowledge Representation (Module 2)

Last update: 26 November 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

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

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

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)` $\land$ `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. $\{\texttt{good}, \texttt{neutral}, \texttt{bad}\}$). 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

Semantic networks

Graphical representation of objects and categories connected through labelled links.

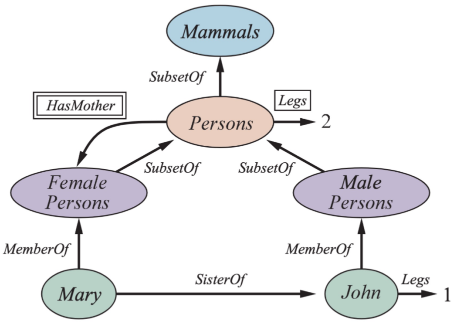

Figure 2.1: Example of semantic network

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

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.

Multiple inheritance reasoning

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

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

Frames

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

- 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 $\mathtt{d}_1 \dots \mathtt{d}_n$ are concepts, then $[\mathtt{AND}\ \mathtt{d}_1 \dots \mathtt{d}_n]$ is a concept.

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

- If $\mathtt{d}_1$ and $\mathtt{d}_2$ are concepts, then $(\mathtt{d}_1 \sqsubseteq \mathtt{d}_2)$ is a sentence.
- If $\mathtt{d}_1$ and $\mathtt{d}_2$ are concepts, then $(\mathtt{d}_1 \doteq \mathtt{d}_2)$ is a sentence.
- If `c` is a constant and `d` is a concept, then $(\mathtt{c} \rightarrow \mathtt{d})$ 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.

Assetion box (A-box) List of facts about individuals. Assetion 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 $\mathtt{d}_1 \dots \mathtt{d}_n$] Individuals belonging to all the categories $\mathtt{d}_1 \dots \mathtt{d}_n$.

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

$\mathtt{d}_1 \sqsubseteq \mathtt{d}_2$ Concept $\mathtt{d}_1$ is subsumed by $\mathtt{d}_2$.

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

$\mathtt{d}_1 \doteq \mathtt{d}_2$ Concept $\mathtt{d}_1$ is equivalent to $\mathtt{d}_2$.

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 $\mathfrak{I}$ 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, $\mathcal{I}[\mathtt{c}] \in \mathcal{D}$.

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

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

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

  [ALL r d]

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

  [EXISTS n r]

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

[FILLS r c]

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

[AND $\texttt{d}_1 \dots \texttt{d}_n$]

$$\mathcal{I}[\texttt{[AND } \texttt{d}_1 \dots \texttt{d}_n\texttt{]}] = \mathcal{I}[\texttt{d}_1] \cap \dots \cap \mathcal{I}[\texttt{d}_n]$$

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

- $\mathfrak{I} \models (\texttt{c} \rightarrow \texttt{d})$ iff $\mathcal{I}[\texttt{c}] \in \mathcal{I}[\texttt{d}]$.
- $\mathfrak{I} \models (\texttt{d}_1 \sqsubseteq \texttt{d}_2)$ iff $\mathcal{I}[\texttt{d}_1] \subseteq \mathcal{I}[\texttt{d}_2]$.
- $\mathfrak{I} \models (\texttt{d}_1 \doteq \texttt{d}_2)$ iff $\mathcal{I}[\texttt{d}_1] = \mathcal{I}[\texttt{d}_2]$.

Given a set of sentences S, $\mathfrak{I}$ models S if $\mathfrak{I} \models S$.

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

$$\forall \mathfrak{I} : (\mathfrak{I} \models S) \rightarrow (\mathfrak{I} \models \alpha)$$

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 $\texttt{d}$ is satisfiable w.r.t. S if: Satisfiability

$$\exists \mathfrak{I}, (\mathfrak{I} \models S) : \mathfrak{I}[\texttt{d}] \neq \varnothing$$

Subsumption A concept $\texttt{d}_1$ is subsumed by $\texttt{d}_2$ w.r.t. S if: Subsumption

$$\forall \mathfrak{I}, (\mathfrak{I} \models S) : \mathfrak{I}[\texttt{d}_1] \subseteq \mathfrak{I}[\texttt{d}_2]$$

Equivalence A concept $\texttt{d}_1$ is equivalent to $\texttt{d}_2$ w.r.t. S if: Equivalence

$$\forall \mathfrak{I}, (\mathfrak{I} \models S) : \mathfrak{I}[\texttt{d}_1] = \mathfrak{I}[\texttt{d}_2]$$

Disjointness A concept $\texttt{d}_1$ is disjoint to $\texttt{d}_2$ w.r.t. S if: Disjointness

$$\forall \mathfrak{I}, (\mathfrak{I} \models S) : \mathfrak{I}[\texttt{d}_1] \neq \mathfrak{I}[\texttt{d}_2]$$

Theorem 3.3.1 (Reduction to subsumption)**.** Given the concepts $\texttt{d}_1$ and $\texttt{d}_2$, it holds that: Reduction to subsumption

- $\texttt{d}_1$ is unsatisfiable $\iff \texttt{d}_1 \sqsubseteq \bot$.
- $\texttt{d}_1 \doteq \texttt{d}_2 \iff \texttt{d}_1 \sqsubseteq \texttt{d}_2 \wedge \texttt{d}_2 \sqsubseteq \texttt{d}_1$.
- $\texttt{d}_1$ and $\texttt{d}_2$ are disjoint $\iff (\texttt{d}_1 \cap \texttt{d}_2) \sqsubseteq \bot$.

3.3.2 A-box reasoning

Given a constant $\texttt{c}$, a concept $\texttt{d}$ and a set of sentences S, we can determine the following:

Satisfiability A constant $\texttt{c}$ satisfies the concept $\texttt{d}$ if: Satisfiability

$$S \models (\texttt{c} \rightarrow \texttt{d})$$

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:

$$KB \models (\texttt{d} \sqsubseteq \texttt{e})$$

The following algorithms can be employed:

Structural matching Structural matching

1. Normalize `d` and `e` into a conjunctive form:

$$\texttt{d} = \texttt{[AND d}_1 \texttt{ ...d}_n\texttt{]} \qquad \texttt{e} = \texttt{[AND e}_1 \texttt{ ...e}_m\texttt{]}$$

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` $\texttt{c}_1$ ... $\texttt{c}_n$`]` Concept only satisfied by $\texttt{c}_1$ ... $\texttt{c}_n$.

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` $\texttt{r}_1$ $\texttt{r}_2$`]` 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` $\texttt{d}_1$ `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 $\bot$).
- 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