Symbolic Artificial Intelligence
Lecture 1: Introduction to Description Logics and Ontologies

Natalia Díaz Rodríguez , PhD

ENSTA Paris, Institute Polytechnique de Paris and INRIA Flowers
flowers.inria.fr http://asr.ensta-paris.fr/ nataliadiaz.github.io
natalia.diaz@ensta-paris.fr
IA301 Logics and Symbolic Artificial Intelligence
https://perso.telecom-paristech.fr/bloch/OptionIA/Logics-SymbolicAI.html

Course summary:

This course aims at providing the bases of symbolic AI, along with a few selected advanced topics. It includes courses on formal logics, ontologies, symbolic learning, typical AI topics such as revision, merging, etc., with illustrations on preference modelling and image understanding.

These 3 units: Ontologies, Knowledge Representation and Reasoning

Skills:

At the end of the course students you will be able to understand different kinds of logic families, formulate reasoning in such formal languages, and manipulate tools to represent knowledge and its adaptation to imprecise and incomplete domains through the use of OWL, Protégé and fuzzyDL.

Prerequisites:

Basic knowledge in computer science and algebra

Symbolic AI: Description Logics and Ontologies

Why study Symbolic AI?

Because:

- Deep learning-based AI is unable to reason, yet
- Neural models are black boxes, hard to interpret
- There is more to predict than what is visible or readable (CV, NLP):
  - Concepts, abstraction, embodiment, ...$\rightarrow$ context
- Eventually, decision support AI systems need to be told what the rules are (policies, ethics, laws) $\rightarrow$ requires knowledge representation (KR) and knowledge reasoning (KR)
  - If inference interpretation is wrong, decisions will be wrong as well
  - *The integration of both data-driven learning and knowledge-driven learning is probably what human learning is all about* [15, 19].

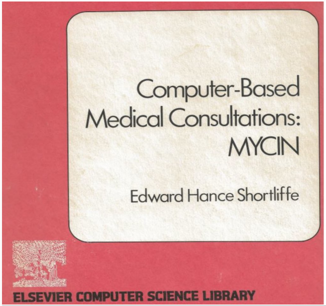
Knowledge Representation (KR)

- Goal: develop formalisms for providing high-level descriptions of the world that can be effectively used to build intelligent applications [3].
- KR languages need a well-defined syntax and a formal, unambiguous semantics -not always true for predecessor KR approaches-:
  - Semantic Networks [Quillian'67] (Semantic Memory Model, labeled directed graph)
  - Frames paradigm [Minsky'74] (A frame represents a concept and is characterized by a number of attributes (*slots*) that members of its class can have)
- High-level descriptions: concentrate on representing relevant aspects for a given application, while ignoring irrelevant details.

Knowledge Representation: The origins

MYCIN [31] (1976): influential in the development of expert systems, esp. rule-based approaches. One of the first programs to create a reasoning network for representing and utilizing judgmental knowledge, model inexact reasoning that typify real-world problems[1].

Later: NELL (Never Ending Language Learning, 2010) [12],...



[1]MYCIN's aim: give advice regarding antimicrobial selection, making it acceptable to physicians. 3 goals: ability to 1) give good **advice**, 2) **explain** the basis for its advice, 3) **acquire** new knowledge easily so advice can improve over time.

Description Logics (DL)

- A family of formal logic-based knowledge representation formalisms tailored towards representing terminological knowledge of a domain in a structured and well-understood way.
- Notions (classes, relations, objects) of the domain are modelled using (atomic) concepts -unary predicates-, (atomic) roles -binary preds-, and individuals to:
  - state *constraints* so that these notions can be interpreted
  - *deduce* consequences (*subclass* and *instance* relationships from definitions and constraints).

Why using DL in Knowledge Representation (KR)...

...rather than general first-order predicate logic?

- Because is a decidable[2] fragment of FOL, therefore, amenable for automated reasoning
- Because generating justifications for entailment[3] is possible[4]
- **Ex.**

A-BOX

human(Aristotle)

T-BOX

human $\sqsubseteq$ mortal

Aristotle $\in$ mortal ?

[2] A logic is decidable if computations/algorithms based on it will terminate in a finite time
[3] R: set of clauses, γ: a ground atom; $R \models \gamma$ if every model satisfying R also satisfies γ
[4] `https://github.com/matthewhorridge/owlexplanation`

Description Logics: ABox, TBox and KB

- TBox (Terminological): The vocabulary used to describe concept hierarchies and roles in the KB (the world's rules, the *schema* in a DB setting). Can contain two kinds of axioms asserting that:
  - An individual is an instance of a given concept
  - A pair of individuals is an instance of a given role [4].
- ABox (Assertional): States properties of individuals in the KB (the data)
- Statements in TBox and ABox can be interpreted with DL rules and axioms[5] to enable reasoning and inference (including satisfiability, subsumption, equivalence, instantiation, disjointness, and consistency).

[5] *Axioms* (logical assertions) together comprise the overall theory that the ontology describes in its domain

Examples TBox concept definitions [4][6]:

- *Men that are married to a doctor and all of whose children are either doctors or professors*: $\texttt{HappyMan} \equiv \texttt{Human} \sqcap \neg\ \texttt{Female} \sqcap (\exists\ \texttt{married.Doctor}) \sqcap (\forall\ \texttt{hasChild.(Doctor} \sqcup \texttt{Professor))}$.
- *Only humans can have human children:* $\exists\ \texttt{hasChild.Human} \sqsubseteq \texttt{Human}$

Ex. ABox:

- $\texttt{HappyMan(BOB)},\ \texttt{hasChild(BOB, MARY)},\ \neg\ \texttt{Doctor(MARY)}$

[6] The variable-free syntax of DL makes TBox statements *easier to read* than the corresponding first-order formulae.

Description Logics: ABox, TBox and KB

- *Number restrictions*: describe the nr of relationships of a particular type that individuals can participate in. **Ex**:
  A person can be married to at most 1 other indiv: $\texttt{Person} \sqsubseteq \leqslant 1\ \texttt{married}$
- *Qualified Nr restrictions*: restrict the type of individuals that are counted by a given number restriction. **Ex**. HappyMan: man that has between 2-4 children:
  $\texttt{HappyMan} \equiv \texttt{Human} \sqcap \neg\ \texttt{Female} \sqcap (\exists\ \texttt{married.Doctor}) \sqcap (\forall\ \texttt{hasChild.(Doctor} \sqcup \texttt{Professor))} \sqcap \geqslant 2\ \texttt{hasChild} \sqcap \leqslant 4\ \texttt{hasChild}.$

Ex. HappyMan: man that has between 2-4 children:

$\texttt{HappyMan} \equiv \texttt{Human} \sqcap \neg\ \texttt{Female} \sqcap (\exists\ \texttt{married.Doctor}) \sqcap (\forall\ \texttt{hasChild.(Doctor} \sqcup \texttt{Professor))} \sqcap \geqslant 2\ \texttt{hasChild} \sqcap \leqslant 4\ \texttt{hasChild}.$

How to modify HappyMan with "has at least 2 children who are doctors"?

Ex. $\texttt{HappyMan}$: man that has between 2-4 children:

$\texttt{HappyMan} \equiv \texttt{Human} \sqcap \neg \texttt{ Female} \sqcap (\exists \texttt{ married.Doctor}) \sqcap (\forall \texttt{ hasChild.(Doctor} \sqcup \texttt{ Professor))} \sqcap \geqslant 2 \texttt{ hasChild} \sqcap \leqslant 4 \texttt{ hasChild.}$

How to modify $\texttt{HappyMan}$ with "has at least 2 children who are doctors"?

$\texttt{HappyMan} \equiv \texttt{Human} \sqcap \neg \texttt{ Female} \sqcap (\exists \texttt{ married.Doctor}) \sqcap (\forall \texttt{ hasChild.(Doctor} \sqcup \texttt{ Professor))} \sqcap \geqslant 2 \texttt{ hasChild.}\textbf{Doctor} \sqcap \leqslant 4 \texttt{ hasChild.}$

Doctor in this case is called a **filler** (a class in this case)

Description Logics: ABox, TBox and KB Examples[7]

What can we do with a Knowledge Base (KB = Ontology + instances)?

A-BOX

man(john)	loves(john,mary)
woman(mary)	loves(mary,sam)
man(sam)	married(sam,sue)
woman(sue)	happy(sam)

Some assertions...

T-BOX

...and some rules:

$bachelor \doteq \neg \exists married.\top \sqcap man$	*„bachelors are unmarried men"*
$married \doteq married^{-1}$	(being married to so. is reflexive)
$\exists married.\top \sqsubseteq happy$	*„all married people are happy"*
$\exists_{\geq 2} love \sqsubseteq \perp$	*„you can love at most one person"*
$\exists married.woman \sqsubseteq \exists love.woman$	*„someone married to a woman also loves a woman"*

[7] [Resources for Comp' Linguists. Regneri & Wolska'07]

Description Logics: ABox, TBox and KB[8]

A Knowledge Base $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox.

An interpretation $\mathcal{I}$ is a model of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{I}$ is a model of $\mathcal{A}$.

$\mathcal{AL}$ (attribute language) logic: the minimal logic with a practically usable vocabulary.

If $\mathcal{A}$ and $\mathcal{B}$: atomic concepts; $\mathcal{C}$ and $\mathcal{D}$: concept descriptions; $\mathcal{R}$: atomic role, semantics defined using interpretation $\mathcal{I}$ consist of:

- non-empty set $\Delta^{\mathcal{I}}$ (the domain of interpretation)
- an interpretation function that assigns:
  a set $\mathcal{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to every atomic concept $\mathcal{A}$
  a binary relation $\mathcal{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to every atomic role $\mathcal{R}$.

Concepts $\mathcal{C}$ and $\mathcal{D}$ are equivalent ($\mathcal{C} \equiv \mathcal{D}$), if $\mathcal{C}^{\mathcal{I}} \equiv \mathcal{D}^{\mathcal{I}}$ for all interpretations $\mathcal{I}$.

[8] http://www.obitko.com/tutorials/ontologies-semantic-web/syntax-and-semantics.html

Description Logics[9]: $\mathcal{AL}$ (Attributive Language) logic syntax and semantics

Syntax	Semantics	Comment
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	atomic concept
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	atomic role
$\top$	$\Delta^{\mathcal{I}}$	top (most general) concept
$\bot$	$\emptyset$	bottom (most specific) concept
$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$	atomic negation
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	intersection
$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\}$	value restriction
$\exists R.\top$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\}$	limited existential quantification

[9] http://www.obitko.com/tutorials/ontologies-semantic-web/syntax-and-semantics.html

Description Logics[11]: $\mathcal{AL}$ logic basic extensions

The name of the logic is formed from the string $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$[10].

Name	Syntax	Semantics	Comment
$\mathcal{U}$	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	union of two concepts
$\mathcal{E}$	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$	full quantification
$\mathcal{N}$	$\geqslant nR$ $\leqslant nR$	$\{a \in \Delta^{\mathcal{I}} \mid \ \lvert\{b \mid (a,b) \in R^{\mathcal{I}}\}\rvert \geq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \ \lvert\{b \mid (a,b) \in R^{\mathcal{I}}\}\rvert \leq n\}$	number restriction
$\mathcal{C}$	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	negation of arbitrary concept

[10] $\mathcal{ALEN}$: $\mathcal{AL}$ extended with full existential quantification and number restrictions

[11] http://www.obitko.com/tutorials/ontologies-semantic-web/syntax-and-semantics.html

Description Logics[12]: $\mathcal{AL}$ logic - extensions of interest

:

- $\mathcal{S}$: role transitivity: `hasAncestor`
- $\mathcal{H}$: role hierarchy: `hasParent` subrole of `hasAncestor`.
- $\mathcal{I}$: role inverse: `hasChild` and `hasParent`
- $\mathcal{F}$: functional role in concept creation
- $\mathcal{O}$: nominals $a_1, ..., a_n$ (concept declared by enumeration)

[12] http://www.obitko.com/tutorials/ontologies-semantic-web/syntax-and-semantics.html

Description Logics Families (increasing comput. complexity):

- $\mathcal{EL}$: A prominent tractable DL
- $\mathcal{ALC}$: A basic DL which corresponds to multi-modal K logic [Schild'91] K_n[13].
- $\mathcal{SHIQ}$: Very expressive DL basis of the OWL family

DL	concept and role expressions	TBox axioms
$\mathcal{EL}_\bot$	$C ::= A \mid \bot \mid C_1 \sqcap C_2 \mid \exists P.C$ $R ::= P$	$C_1 \sqsubseteq C_2$
$\mathcal{ALC}$	$C ::= A \mid C_1 \sqcap C_2 \mid \neg C \mid \exists P.C$ $R ::= P$	$C_1 \sqsubseteq C_2$
$\mathcal{SHIQ}$	$C ::= A \mid \neg C \mid C_1 \sqcap C_2 \mid (\geq n\, R\, C)$ $R ::= P \mid P^-$	$C_1 \sqsubseteq C_2$ $R_1 \sqsubseteq R_2$ $\text{Trans}(R)$

[13] Important extensions: inverse roles, number restrictions, and concrete domains.

Description Logics Applications

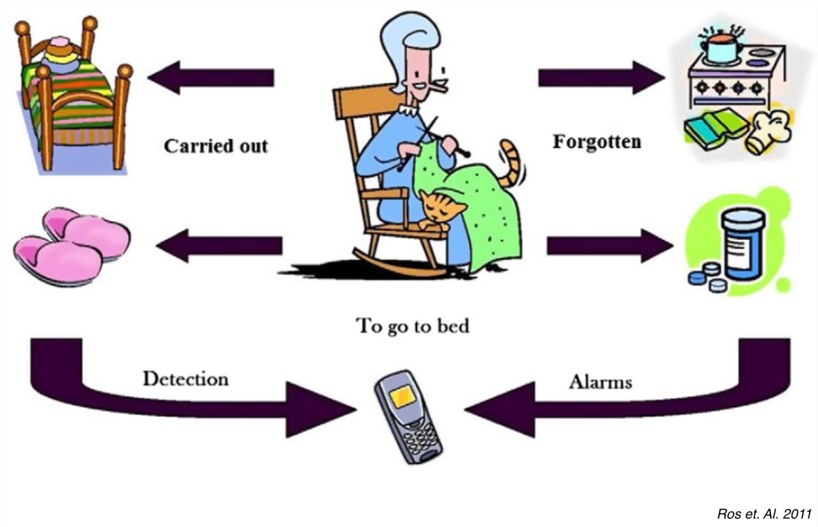
- NLP, DB, and biomedicine[14], healthcare (activity recognition [21, 20], lifestyle profiling [18, 22], rehabilitation [23]), fashion [9, 8],...
- Most notable success: adoption of DL-based OWL as SW std[15].

Why adopting DLs as ontology languages?

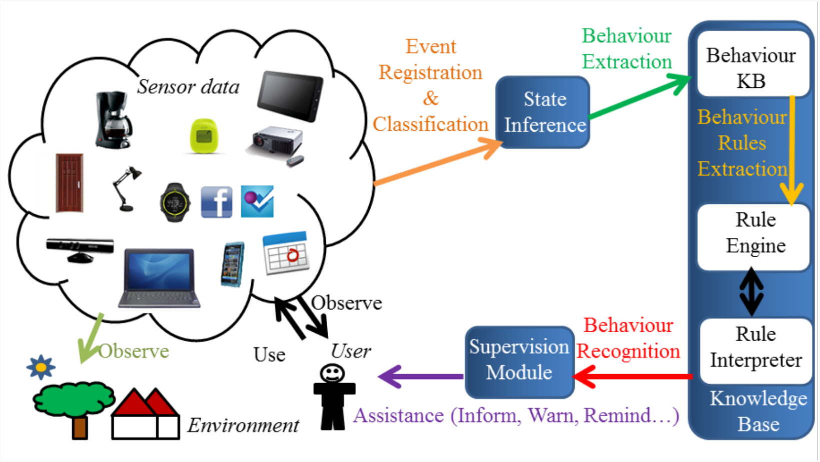
- For a formal, unambiguous semantics of FOL easy to describe and comprehend
- To provide *expressiveness* for constructing concepts and roles, *constraining* their interpretations and instantiating concepts and roles with individuals;
- To provide optimized *inference* procedures (deducing *implicit* knowledge from *explicit* one).

[14] geneontology.org
[15] http://www.w3.org/TR/owl-features/

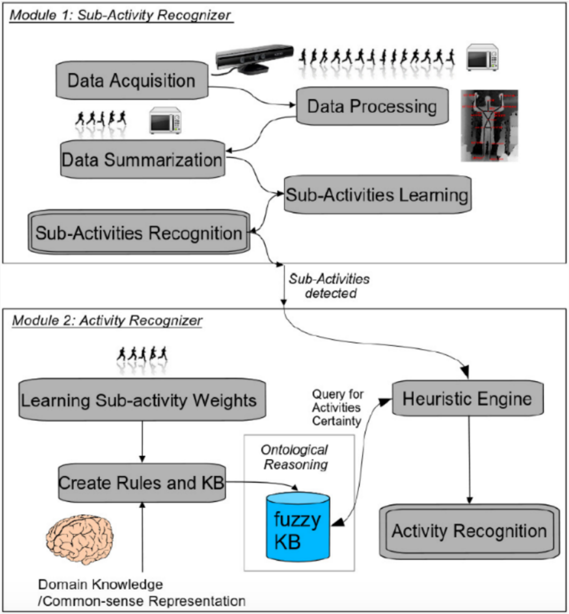
Description Logics Applications: Human activity recognition (HAR)[17]



Description Logics Applications: Human activity recognition [17]



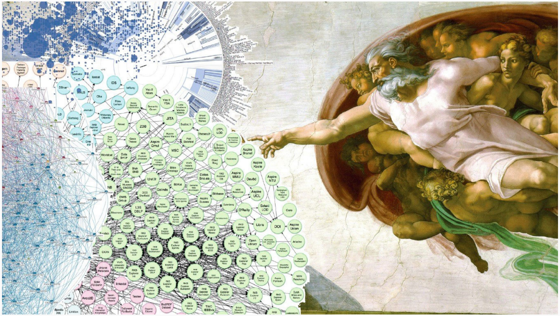
Description Logics Applications: HAR: the big picture [17]



The Semantic Web

The Semantic Web (SW) [5][16]

- *An extension of the web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*
- W3C standard for defining data on the Web.
- XML tags conform to RDF and OWL formats.
- Refers to *things* in the world as resources



[16] http://www.cs.rpi.edu/academics/courses/fall07/semantic/CH1.pdf

RDF: The Resource Description Framework

- Set of tools that use concepts from *graph theory* to add relationships and semantics to *unstructured* data such as the WWW.
- Aim: machine interoperation of cross-domain data and merging info. from different sources as effortless as possible.
- RDF *triple*: foundation of the RDF data model: a subject, predicate and object resource that forms a statement. Triples consisting of matching subjects and objects can be linked together via an *RDF graph* hosted in an RDF store.
- SPARQL[17]: W3C std query language for RDF.

[17] '*sparkle*', SPARQL: Simple Protocol and RDF Query Language

RDF example: Namespaces, URIs and Identity[19]

RDFS: RDF Schema, vocabulary[18]

QUESTION?: How to know when a node in one graph the same as a node in another graph?

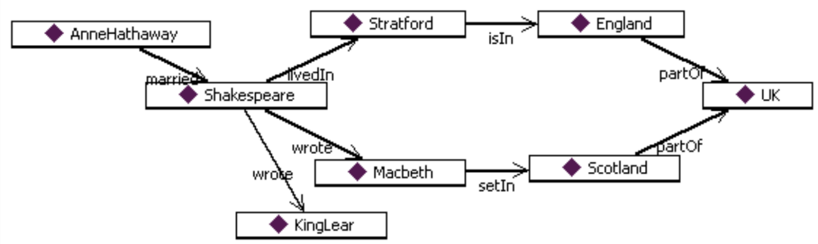
Subject	Predicate	Object
Shakespeare	Wrote	King Lear
Shakespeare	Wrote	Macbeth
Anne Hathaway	Married	Shakespeare
Shakespeare	Lived In	Stratford
Stratford	Is in	England
Macbeth	Set in	Scotland
England	Part of	The UK
Scotland	Part of	The UK

[18] *Intensional* (logic): Allows distinct entities with the same extension (not extensional). *Extensional* (logic): A set-based theory or logic of classes, in which classes are considered to be sets, properties considered to be sets of <object, value> pairs, and so on. A theory which admits no distinction between entities with the same extension.

[19] http://www.cs.rpi.edu/academics/courses/fall07/semantic/CH3.pdf

RDF example: Namespaces, URIs and Identity[20]

When they share the Uniform Resource Identifier (URI) in RDF.



[20] http://www.cs.rpi.edu/academics/courses/fall07/semantic/CH3.pdf

Reasoning (Rule) Engine[24]

→ software able to infer logical consequences from asserted facts/ axioms

Logic Programming:

- Backward chaining[21]
- From goal to facts, applying rules backwards
- Conservative
- Unification[22].
- Backtracking

Rule-based (Prod. Rule) Systems:

- Forward chaining[23]
- Facts activate rules that generate new facts
- Potentially destructive
- Pattern matching
- Parallelism

[21] To test if $R \models \gamma$, we work backwards from γ, looking for rules in R whose head unifies with γ. Tree root: node containing γ; search terminates when a node with no atoms remaining to be proved [25] is found.

[22] Solves equations among symbolic expressions by computing a complete and minimal substitution set covering all solutions and no redundant members.

[23] To test if $R \models \gamma$, we check if $\gamma \in \mathit{consequences}(R)$ [25].

[24] [Sistemi a Regole di Produzione, S. Bragaglia'13]

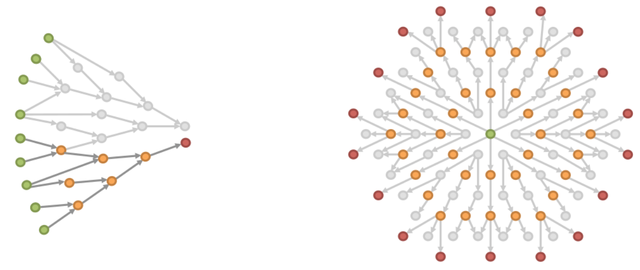
Logic Programming VS Rule-based Systems (Production rules)[25]:

Backward vs Forward chaining - at the start:

[25] [Sistemi a Regole di Produzione, S. Bragaglia'13]

Logic Programming VS Rule-based Systems (Production rules):[26]:

Backward vs Forward chaining - at the end:



[26] [Sistemi a Regole di Produzione, S. Bragaglia'13]

What can a DL reasoner[31] do?

More than classification!: Discover (infer) implicit information (e.g., using necessary and sufficient conditions. **Ex**. `CheesyPizza`)

- (Class) Consistency checking (**Ex**.: `MeatyVegetableTopping`)[27] and Equivalence checking
- Instantiation checking (e.g., determine *domain* and *fillers* of a role[28])
- Retrieval tasks: all individuals of a concept, all concepts of an individual
- Subsumption checking (compute classification hierarchy, find parent concepts[29], predecessors[30] (/successors). **Ex**. "Are *cities* `locations?`")

[27] In Protégé inconsistent classes turn red (cannot possibly contain any individual)
[28] Fillers of R: all f s.t. $\exists x.R(x, f)$
[29] Parents of C: the most specific C' s.t. $C \sqsubseteq C'$ (children analogously)
[30] Predecessors of C: all C' s.t. $C \sqsubseteq^* C'$ (successors analogously)
[31] Ex. reasoners: **Pellet**, RACER, FaCT, DROOLs. Rule (engine) production systems: JBoss Drools, OPS5, CLIPS, Jess, ILOG, JRules, BizTalk.

So, What can a DL reasoner do? e.g., RACER[32]

Example queries:

Is Sue happy?
(Does ‚happy‘ contain Sue?)

Can Mary love John?
(loves(mary, john) -> consistent?)

What properties does Mary have?
(Concepts containing mary)

A-BOX

man(john)	loves(john,mary)
woman(mary)	loves(mary,sam)
man(sam)	married(sam,sue)
woman(sue)	happy(sam)

T-BOX

$\text{bachelor} \doteq \neg \exists \text{married}.\top \sqcap \text{man}$

$\text{married} \doteq \text{married}^{-1}$

$\exists \text{married}.\top \sqsubseteq \text{happy}$

$\exists_{\geq 2} \text{love} \sqsubseteq \bot$

$\exists \text{married.woman} \sqsubseteq \exists \text{love.woman}$

32[Resources for Comp' Linguists. Regneri & Wolska'07]

Common Operators in Description Logics [2]

Constructor	Syntax	Semantics
concept name	C	$C^{\mathcal{I}}$
top	$\top$	$\Delta^{\mathcal{I}}$
negation ($\mathcal{C}$)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
disjunction ($\mathcal{U}$)	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
universal quant.	$\forall R.C$	$\{d_1 \mid \forall d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \rightarrow d_2 \in C^{\mathcal{I}})\}$
existential quant. ($\mathcal{E}$)	$\exists R.C$	$\{d_1 \mid \exists d_2 \in \Delta^{\mathcal{I}}.(R^{\mathcal{I}}(d_1, d_2) \wedge d_2 \in C^{\mathcal{I}})\}$
number restr. ($\mathcal{N}$)	$(\geq n\ R)$	$\{d_1 \mid \lvert\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\rvert \geq n\}$
	$(\leq n\ R)$	$\{d_1 \mid \lvert\{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\}\rvert \leq n\}$
one-of ($\mathcal{O}$)	$\{a_1, \ldots, a_n\}$	$\{d \mid d = a_i^{\mathcal{I}}$ for some $a_i\}$
role filler ($\mathcal{B}$)	$\exists R.\{a\}$	$\{d \mid R^{\mathcal{I}}(d, a^{\mathcal{I}})\}$
role name	R	$R^{\mathcal{I}}$
role conjunction ($\mathcal{R}$)	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
inverse roles ($\mathcal{I}$)	R^{-1}	$\{(d_1, d_2) \mid R^{\mathcal{I}}(d_2, d_1)\}$

Reasoning tasks [2]

- $\{C_1, C_2, ...\}$ atomic concepts
- $\{R_1, R_2 ...\}$ atomic roles
- $\{a_1, a_2, ...\}$ individuals
- Σ a Knowledge Base (KB)

– *Subsumption*, $\Sigma \models C_1 \sqsubseteq C_2$.
  Check whether for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
– *Instance Checking*, $\Sigma \models a : C$.
  Check whether for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
– *Relation Checking*, $\sigma \models (a, b) : R$.
  Check whether for all interpretations $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.
– *Concept Consistency*, $\Sigma \not\models C \doteq \bot$.
  Check whether for some interpretation $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$ we have $C^{\mathcal{I}} \neq \{\}$.
– *Knowledge Base Consistency*, $\Sigma \not\models \bot$.
  Check whether there exists $\mathcal{I}$ such that $\mathcal{I} \models \Sigma$.

Logical Reasoning Capabilities: Subsumption

The task of computing the task hierarchy (*is-a* super/sub class relationship):

- *A* subsumes *B* if *A* is a superclass of *B*
- Defined explicitly (*asserted*), or *inferred* by a reasoner
- Superclass of all OWL Classes: `owl:Thing`

Reasoning tasks for concepts [3]

Satisfiability: A concept C is *satisfiable* with respect to $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. In this case we say also that $\mathcal{I}$ is a *model* of C.

Subsumption: A concept C is *subsumed* by a concept D with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.

Equivalence: Two concepts C and D are *equivalent* with respect to $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In this case we write $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.

Disjointness: Two concepts C and D are *disjoint* with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of $\mathcal{T}$.

The Curry-Howard-Voevodsky correspondence [33]

type theory	set theory	logic	homotopy theory
A	set	proposition	space
$x : A$	element	proof	point
$\emptyset, 1$	$\emptyset, \{\emptyset\}$	$\bot, \top$	$\emptyset, *$
$A \times B$	set of pairs	A and B	product space
$A + B$	disjoint union	A or B	coproduct
$A \to B$	set of functions	A implies B	function space
$x : A \vdash B(x)$	family of sets	predicate	fibration
$x : A \vdash b : B(x)$	fam. of elements	conditional proof	section
$\prod_{x:A} B(x)$	product	$\forall x.B(x)$	space of sections
$\sum_{x:A} B(x)$	disjoint sum	$\exists x.B(x)$	total space
$p : x =_A y$	$x = y$	proof of equality	path from x to y
$\sum_{x,y:A} x =_A y$	diagonal	equality relation	path space for A

[33] [Riehl'18] `http://www.math.jhu.edu/~eriehl/Voevodsky.pdf`

Ontologies

Ontologies

In *Philosophy*: (*Ontological*) Concerned with what kinds of things really exist [Parmenides: not only what exists, but what can exist].

In *AI*: *A explicit (formal) specification of a (shared) conceptualization* [26, 10]; defines concepts, individuals, relationships and constraints (functions, attributes) within a domain.

Why Ontologies?

- The power of representation (separate declarative & procedural knowledge)
- Logical reasoning capabilities: deduction, abduction, and subsumption
- Explainability: to extract a minimal set of covering models of interpretation from a KB based on a set of observed actions, which could explain the observations [14].
- To represent and share knowledge by using a common vocabulary
- To promote interoperability, knowledge reuse, and info. integration with automatic validation

Ontologies

- **Facilitate** KB *modularity* [6], allow machine-readability by agents [24]
- Among semantic technologies, the most used formalism to represent and reason with knowledge.
- **Applications**: Information retrieval, search, question answering, m-Government emergency response services [1] or detecting information system conflicts [28]
  $\rightarrow$ and transport infraction detection in Paris!

Semantic Web (SW) Family of Languages [33]

3 main streams:[34]:

- Triple languages (RDF, RDFS). **Ex**. RDF:
  Subject Predicate Object
  ```
  metro:item0 rdf:type metro:Metro
  metro:item0 dc:title ‘‘Allen Station’’
  metro:item0 simile:address "395 N. Allen Av., Pasadena 91106"
  ```
- Ontology (conceptual) languages (OWL2): family that relates to DLs
- Rule-based languages (SWRL[35], RIF[36]). **Ex**. RIF:
  ```
  ForAll ?Buyer ?Item ?Seller
  buy(?Buyer ?Item ?Seller) :- sell(?Seller ?Item ?Buyer)
  ```

---

[34] http://www.umbertostraccia.it/cs/download/papers/SUM11/SUMSlidesStraccia11.pdf
[35] Semantic Web Rule Lang.: High-level abstract syntax for Horn-like rules in both OWL DL and OWL Lite sub-languages of OWL.
[36] Rule Interchange Format, family relating to the Logic Programming (LP) paradigm [32][11])

Web Ontology Language (OWL)

- W3C std based on the KR formalism of DL [4]
  - Most used language to model formal ontologies
  - DL reasoning supports incremental inference
- Models concepts, roles and individuals.
  - Concepts: define aggregation of things
  - Individuals: instances of concepts
  - Properties (relationships): link individuals from the domain to individuals from the range

OWL Properties Restrictions:

→ *Anonymous* class definitions that group individuals together based on at least one object prop.

Ex.: "class of individuals that have at least one `hasTopping` relationship to individuals member of `MozzarellaTopping`".

- Existential restrictions (∃): An individual of the class *Pizza* must have (**at least one**) `PizzaBase` (`owl:SomeValuesFrom` restriction)[37]:
  `Pizza and hasBase` *`some`* `PizzaBase`
  Should paraphrase: "Among other things..."
- Universal Restriction (∀): individuals from the class `VegetarianPizza` can **only** have toppings that are vegetarian toppings. (`owl:AllValuesFrom` restriction).
  `Pizza and hasTopping` *`only`* `VegetarianTopping`
  Should paraphrase: "All and only values from"
- Necessary conditions: {Class} ⇒ {[conditions]} (called superclasses, *Subclass Of* Protégé slot)
- Necessary and sufficient conditions: {Class} ⇔ {[conditions]} (called equivalent classes, *Equivalent To* Protégé slot)

[37] https://www.cambridgesemantics.com/blog/semantic-university/learn-owl-rdfs/owl-references-humans/

Semantic Web Family of Languages [33]

Conceptual languages (OWL, OWL 2) and OWL 2 **profiles**:

- OWL EL: instance/subsumption checking decided in polynomial time. Useful: large size of properties and/or classes.
- OWL QL: (relates to the DL family DL-Lite): Useful: very large instance data volumes[38].
- OWL RL[39] Useful for scalable reasoning without sacrificing much expressive power.

[38] conjunctive query answering via query rewriting and SQL
[39] Maps to Datalog, same complexity: polyn. in size of the data, exp. t., wrt. KB size

Web Ontology Language (OWL) [7]

OWL comprises 3 **sub-languages**[40] of increasing expressive power (all sublanguages of OWL2-DL, as itself, tractable):

- OWL Lite: Lowest complexity (only 0/1 card. constr., no disjointness nor enumerated classes).
- OWL DL: (based on DL, OWL DL $\subseteq$ OWL Full): Decidable, permits inconsistency checking
- OWL Full: Max. expressiveness with syntactic freedom of RDF[41]

Which sub-language to use?[42]

- *Are OWL-Lite constructs sufficient? OWL-DL vs OWL-Full?*
- Prioritize: Carrying out automated reasoning vs using highly expressive and powerful modelling (e.g. classes of classes)?

[40] **Our focus: OWL 2** and **OWL DL**.

[41] When expressiveness is more important than being able to guarantee the decidability /computational completeness/ complete reasoning of the language

[42] See `http://www.cs.rpi.edu/academics/courses/fall07/semantic/CH3.pdf` and comparative table `https://ragrawal.wordpress.com/2007/02/20/difference-between-owl-lite-dl-and-full/` and `http://www2.cs.man.ac.uk/~raym8/comp38212/main/node187.html`

OWL constructors and axioms

Constructor	DL Syntax	Example
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	$\mathsf{Human} \sqcap \mathsf{Male}$
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$\mathsf{Doctor} \sqcup \mathsf{Lawyer}$
`complementOf`	$\neg C$	$\neg\mathsf{Male}$
`oneOf`	$\{x_1 \ldots x_n\}$	$\{\mathsf{john}, \mathsf{mary}\}$
`allValuesFrom`	$\forall P.C$	$\forall\mathsf{hasChild}.\mathsf{Doctor}$
`someValuesFrom`	$\exists r.C$	$\exists\mathsf{hasChild}.\mathsf{Lawyer}$
`hasValue`	$\exists r.\{x\}$	$\exists\mathsf{citizenOf}.\{\mathsf{USA}\}$
`minCardinality`	$(\geqslant n\ r)$	$(\geqslant 2\ \mathsf{hasChild})$
`maxCardinality`	$(\leqslant n\ r)$	$(\leqslant 1\ \mathsf{hasChild})$
`inverseOf`	r^-	$\mathsf{hasChild}^-$

OWL constructors and axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
sameAs	$\{x_1\} \equiv \{x_2\}$	{Pres_Bush} $\equiv$ {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
TransitiveProperty	P transitive role	hasAncestor is a transitive role
FunctionalProperty	$\top \sqsubseteq (\leqslant 1\ P)$	$\top \sqsubseteq (\leqslant 1$ hasMother)
InverseFunctionalProperty	$\top \sqsubseteq (\leqslant 1\ P^-)$	$\top \sqsubseteq (\leqslant 1$ isMotherOf$^-$)
SymmetricProperty	$P \equiv P^-$	isSiblingOf $\equiv$ isSiblingOf$^-$

Learning OWL through Protégé examples

Reminder: Why building an ontology?[30]

1. To share common understanding of the info. structure among people/ agents
2. To enable reuse of domain knowledge
3. To make domain assumptions explicit
4. To separate domain knowledge from the operational knowledge
5. To analyze domain knowledge

What does it mean "developing" an ontology?[30]

1. Defining classes in the ontology
2. Arranging them in a taxonomic hierarchy
3. Refining slots and describing its allowed values, filling in the values for slots for instances.

→ 1st step: Determining domain and scope!

Protégé

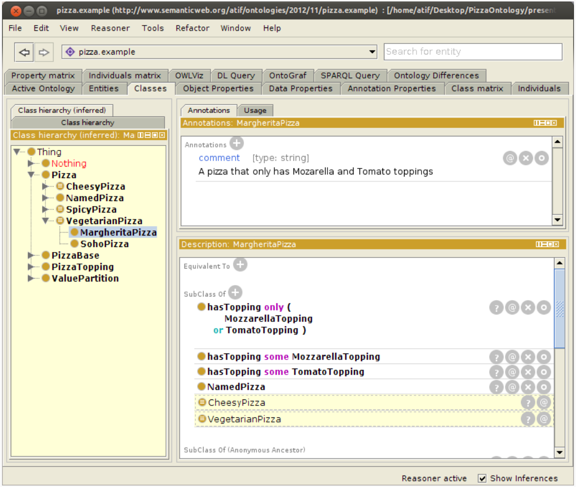
A useful ontology IDE for managing large ontologies and discovering existing ones

- edit
- visualize
- validate KBs

Download: `https://protege.stanford.edu/`

Terminology: OWL Property & Concept Restrictions

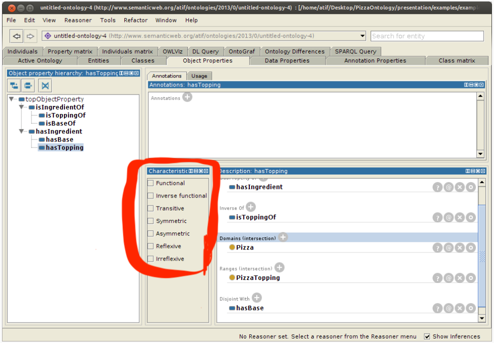
- *Inverse* (object) property: *a pizza has a topping of anchovies* ≡ anchovies is a topping of a pizza
- *Disjoint* concepts: `Calzone` and `Napolitana`. `PizzaTopping` and `PizzaBase`.



Terminology: OWL Property Restrictions

OWL primitives to enrich property definitions. Can you think of examples of ...?:

- *Functional*: `hasAge(A, x)`, `hasBirthMother(A,B)`
- *Inverse functional*: `isBirthModerOf(A,B)`
- *Transitive*: `hasAncestor(A,B)`, `containsIngredient(A,B)`
- *Symmetric*: `married(A, B)` *Anti-symmetric*: `hasFavouriteFlavor(A,B)`
- *Reflexive*: `preparesBreakfast(A, A)`, `dresses(A,A)`
- *Irreflexive*: `isMotherOf(A, B)`



OWL Property Restrictions Exercise: The Simpsons!

	Irreflexive	Asymmetric	Symmetric	Transitive
hasRelationshipTo				
hasSibling				
hasBrother				
hasSister				
hasParent				
hasMother				
hasFather				
hasAunt				
hasUncle				
hasChild				
hasSon				
hasDaugther				
hasGrandParent				
hasSpouse				
hasHusband				
hasWife				

OWL Property Restrictions Exercise: The Simpsons!

See wikipedia for explanations of the characteristics asymmetric[3] , reflexive and irreflexive[4] .

Assuming all relationships have Person as both domain and range, the following is an arguably common interpretation of their characteristics.

	Irreflexive	Asymmetric	Symmetric	Transitive
hasRelationshipTo	X		X	
hasSibling	x		X	X
hasBrother	x			X
hasSister	x			X
hasParent	x	X		
hasMother	x	X		
hasFather	x	X		
hasAunt	x	X		
hasUncle	x	X		
hasChild	x			
hasSon	x			
hasDaugther	x			
hasGrandParent	x			
hasSpouse	x		X	
hasHusband	x	X †)		
hasWife	x	X †)		

Notes:

- x: hasRelationshipTo is irreflexive, so all subproperties of it must also be.
- †) Assuming heteronormativity.

Now, find some missing crosses, yet to be filled in this solution! ;) Which ones are missing?

OWL Property Restrictions Exercise: The Simpsons!

See wikipedia for explanations of the characteristics asymmetric[3] , reflexive and irreflexive[4] .

Assuming all relationships have Person as both domain and range, the following is an arguably common interpretation of their characteristics.

	Irreflexive	Asymmetric	Symmetric	Transitive
hasRelationshipTo	X		X	
hasSibling	x		X	X
hasBrother	x			X
hasSister	x			X
hasParent	x	X		
hasMother	x	X		
hasFather	x	X		
hasAunt	x	X		
hasUncle	x	X		
hasChild	x			
hasSon	x			
hasDaugther	x			
hasGrandParent	x			
hasSpouse	x		X	
hasHusband	x	X †)		
hasWife	x	X †)		

Notes:

- x: hasRelationshipTo is irreflexive, so all subproperties of it must also be.
- †) Assuming heteronormativity.

Now, find some missing crosses, yet to be filled in this solution! ;) Which ones are missing? `hasChild`, `hasSon`, `hasDaughter`, `hasGrandParent` should be asymmetrical.

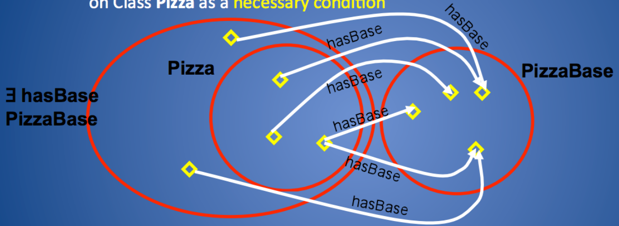
Key to Remember! A simple modelling pipeline

- Start building disjoint tree of primitive concepts. Recall:
  - Classes: Asserted vs Inferred (Pre/post reasoner)
  - Primitive class: Only has necessary conditions, i.e., superclasses.
  - Defined class[43]: has necessary and sufficient conditions, i.e., equivalent classes (**Ex**. `Parent`: the set of all persons that have at least one child).
- (Most often) asserting *polyhierarchies* is bad
  → let the reasoner do it!
  Ex.: `CheesyPizza`: can be `VegetarianPizza`, `SpicyPizza`.
  1. Asserting subclass manually: We lose some encapsulation of knowledge and self-explanation (*Why is this class a subclass of that one?*)
  2. Difficult to maintain (all subclasses may need to be updated)

[43] *Declares* the named class to be equivalent to the *anonymous* class (≡ sign in Protégé interface):
https://protegewiki.stanford.edu/wiki/ProtegeOWL_API_Advanced_Class_Definitions

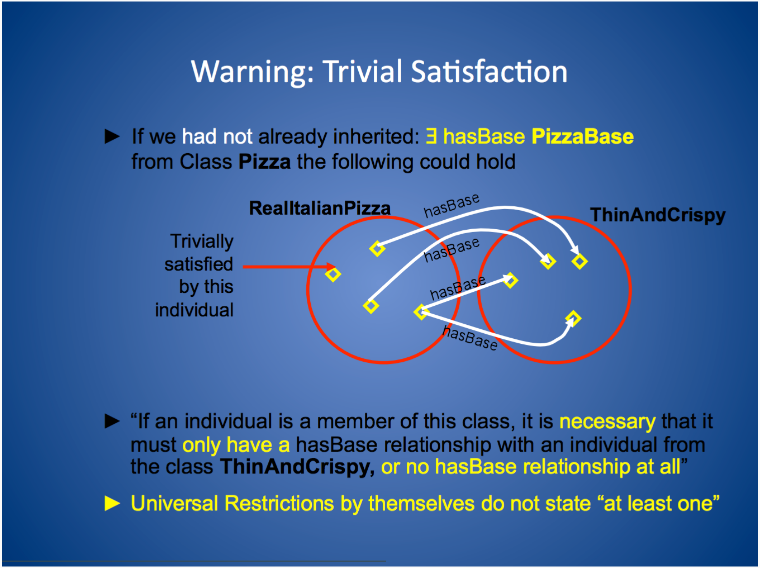
Why? Necessary conditions

- We have created a restriction: ∃ hasBase **PizzaBase** on Class **Pizza** as a necessary condition



- Each necessary condition on a class is a superclass of that class
- ie The restriction ∃ hasBase **PizzaBase** is a superclass of **Pizza**
- As **Pizza** is a subclass of the restriction, all **Pizza**s must satisfy the restriction that they have at least one base from **PizzaBase**

[44] [C. Lagoze, Cornell]



45[C. Lagoze, Cornell]

Take home message: A simple knowledge engineering methodology [30]

- There is no single *correct* way to model a domain ontology, or design methodology[46]
  → depends on application and future extensions
- Concepts in the ontology should be close to objects (*physical* or *logical*)
- Ontology development: necessarily *iterative*

[46] but many ideas + good practices found useful from experience

Logical Reasoning Capabilities: Classification and Disjointness!

Detecting inconsistencies in DL (unsatisfiable axioms):

- OWL assumes that classes overlap! → means an individual could be both a `MeatTopping` and a `VegetableTopping` at the same time!
  → We must state disjointness explicitly in the interface

Open vs Closed World Assumption in ML

A Closed World Assumption "closes" the interpretation by assuming that every fact not explicitly stated to be true is actually **false**.

Open World Assumption (OWA)

What it means: missing information is **not** confirmation of negation. Must state that a description is **complete** (we need `closure` for the given property).

Ex. `MargheritaPizza` toppings must be explicitly limited to their toppings:

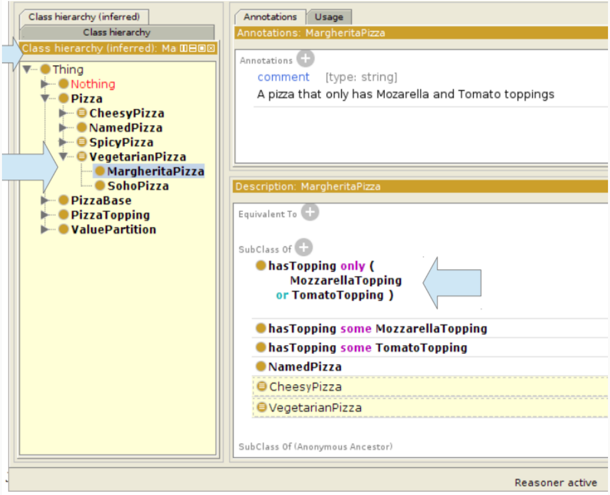
`MargheritaPizza: hasTopping only (MozzarellaTopping or TomatoTopping)`

All `MargheritaPizzas` must have:

- at least 1 topping from `MozzarellaTopping` (Existential restr.)
- at least 1 topping from `TomatoTopping`
- **only** toppings from `MozzarellaTopping` or `TomatoTopping` → no other toppings; The union **closes** the `hasTopping` property on `MargheritaPizza`

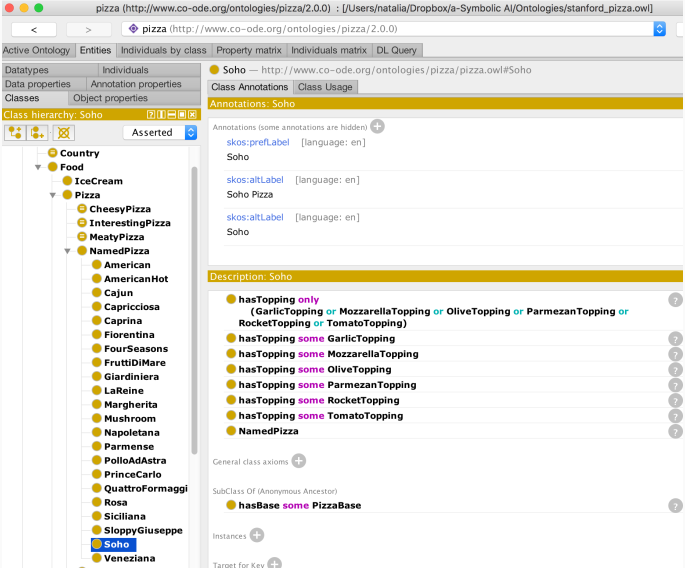
OWA and Universal Restrictions in Protégé

SohoPizza and MargheritaPizza must be explicitly limited to their toppings



Open World Assumption (OWA): Inferring VegetarianPizzas

SohoPizza and MargheritaPizza must be explicitly limited to their toppings so they can be classified as vegetarian pizzas!

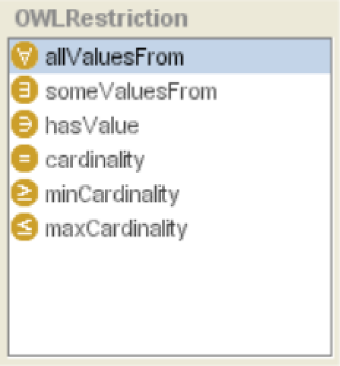


Existential Restrictions vs Universal Restrictions: In Protégé

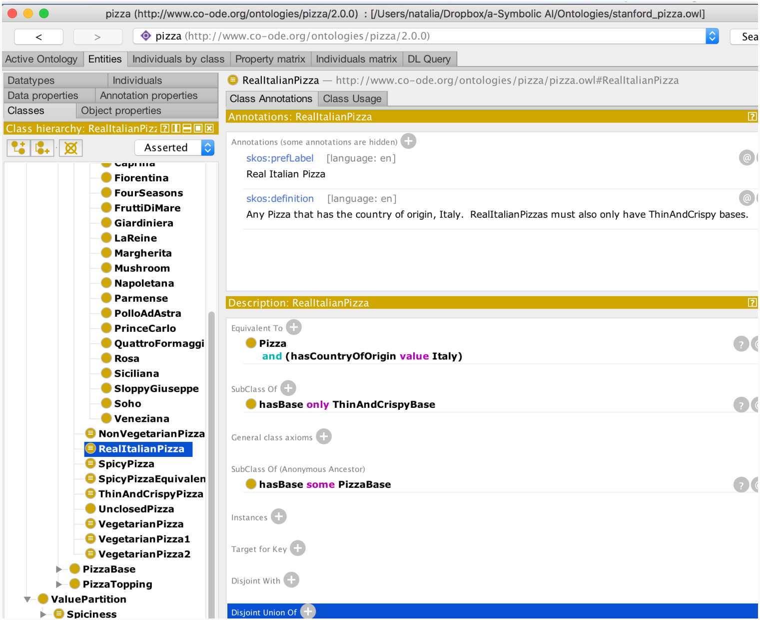
Existential (∃) Restrictions (*some* keyword). ["Among other things..."]

Universal (∀) Restrictions (*only* keyword). [“All and only values from”]

→ Both restrictions added same way (but different restriction type):



Univ. Restr: `RealItalianPizzas` only have bases that are `ThinAndCrispy`



Ontology Modelling Concepts [27]

- *Anonymous class*: all types of restrictions describe an unnamed set that could contain some individuals. When we describe a named class using restrictions, what we are effectively doing is describing anonymous superclasses of the named class.
- *Defined class*: one that is defined with necessary and sufficient conditions (in Equivalent classes slot in Protégé; it can be made by clicking on a class defining a necessary condition and 'Convert to defined class').
- *Existential restrictions*: do not constrain the property relationship to members of the class Class A, it just states that every individual must have at least one prop. relationship with a member of Class A -this is OWA.
- *Universal restrictions*: do not *guarantee* the existence of a relationship for a given property; Existentials, do.
- *Qualified Cardinality Restrictions* (QCR): more strict than QR: they state the class of objects within the restriction [`hasTopping min 3` is the same as `hasTopping min 3 Thing`. QCR: `hasTopping exactly 4 CheeseTopping`]
- *Covering axiom*: the partition of subclasses is complete (e.g. `PizzaSpiciness`)

Ontology Modelling Concepts [27]

- *Data literal*: the character representation of a datatype value, e.g. "Natalia", 25, 28.08
- *UNA* (Unique Name Assumption): OWL does not use UNA; this means that different names may refer to the same individual. Cardinality restrictions rely on *counting* distinct individuals, therefore it is important to **specify** that either "Matt" and "Matthew" are the **same individual**.
- *Explanations*: remember to use them!
  Check the 'Danger' traffic sign in Protégé to get hints on why your model may be wrong. Yes, if no useful message, a restart of Protégé will help (also at any time when you have observed your inconsistency classes in Protégé in red).

Ontology Modelling and patterns [27]

- When 2 ways of modelling a concept: preferably **keep definitions** that are **less verbose** and express same meaning through definitions, than generating extra classes: *InterestingPizza: Pizza and (hasTopping min 3)*
- Specifying a relationship among a class and an individual (`hasValue` construct): `MozzarelaTopping hasCountryOfOrigin value Italy`. Note: With current reasoners the classification is not complete for individuals (`Country` ≡ `America, England, France, Germany, Italy` isn't complete list but meets ontology needs). **Use individuals** in class descriptions **with care**. Enumerated class: anonymous class that lists the specific individuals and only the individuals that it contains (`WeekDay`) (We can attach these individuals to a named class by creating the enumeration as an equivalent class.)
- It's particularly unusual (probably an error), if when describing a class, a universal restriction along a given property is used without **using** a *corresponding* **existential restriction** along the same property.

Homework: By next week:

1. Install Protégé (5.2 or 5.5 Beta, avoid WebProtégé until you consider yourself a Protégé expert ;))[47] In the lab, run in the terminal "`protege`".
2. Find a pair! Think of a problem worth working on that requires an ontology
3. Protégé *Getting Started* and Protégé for *Pizzas in 10 min*[48]
4. Read THE Protégé Tutorial[49]. In the same page you can download the Pizza ontology[50] to play around with it at the same time.
5. Curious to learn more? Play with/extend some fun ontology (Wine [13][51] or Beer[52] ontologies :)) → **When in doubt**: *Ontology development 101: A guide to creating your first ontology*[53][30]. **When stuck**, see [54].

[47] Follow instructions from `https://protege.stanford.edu/` (if asked, choose version with Java Virtual Machine), If problems, see `https://tinyurl.com/ycs5msue`

[48] `https://protegewiki.stanford.edu/wiki/Protege4GettingStarted` and `https://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes`

[49] `http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf`

[50] `http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/`

[51] `https://github.com/NataliaDiaz/Ontologies/blob/master/DidacticOntologies/FuzzyWineOntologyAppCarlsson10/Wine_ontology2.5.owl`

[52] `https://www.cs.umd.edu/projects/plus/SHOE/onts/beer1.0.html`

[53] `https://protege.stanford.edu/publications/ontology_development/ontology101.pdf`

[54] `http://www.cs.cornell.edu/courses/cs431/2008sp/Lectures/public/lecture-4-09-08.pdf`

Searching for stage/internship/superproject/PRe/PFE?

If interested in deep learning, reinforcement learning, symbolic AI, computer vision and NLP for

- robotics and autonomous systems, e.g., self-driving vehicles, drones...
- eXplainable AI
- AI for health
- AI for social good (technology for the blind, computer vision and natural language processing)
- Reinforcement learning
- Examples of project & communities: `www.ContinualAI.org` & `https://lazarilloproject.github.io/`

consider ENSTA Paris U2IS Lab:

- `flowers.inria.fr`
- `http://asr.ensta-paris.fr/`
- `nataliadiaz.github.io`

Send single pdf including grades, CV, and link to your Linkedin and Github: natalia.diaz@ensta-paris.fr

Lewis Carroll

'How do you know I'm mad?' said Alice.
'You must be,' said the Cat,
'or you wouldn't have come here.'

from "Alice's Adventures in Wonderland," Lewis Carroll

Useful Links

1. W3C Glossary[55]
2. MIRO – Minimum Information for Reporting of an Ontology guidelines: a community-validated set of recommendations on what should be reported about an ontology and its development, most importantly in the context of ontology description papers intended for publishing in scientific journals or conferences [29]
3. THE Protégé Tutorial[56]
4. Building OWL Ontologies with Protégé. CS431 –Cornell Univ. 2008 C. Lagoze[57]
5. Resources for Comp‘ Linguists 07 Description Logics - M. Regneri & M. Wolska[58]
6. Tutorial on description logics. I. Horrocks and U. Sattler[59]
7. Probabilistic Logic Programming Languages, F. Riguzzi,[60]
8. Common Pitfalls creating ontologies[61]

9. Building OWL Ontologies with Protégé CS431 –Cornell University, 2008 C. Lagoze[62]
10. Ontology Engineering Methodologies (Ch. 9) [16][63]
11. Resources for Comp' Linguists 07 Description Logics - M. Regneri & M. Wolska[64]
12. An introduction to Ontology Engineering. M. Keet[65].
13. Description Logic, Semantic Web and Ontology Development, S.Bragaglia[66]

[55] https://www.w3.org/TR/rdf-mt/#glossIntensional
[56] http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf
[57] www.cs.cornell.edu/courses/cs431/2008sp/Lectures/public/lecture-4-09-08.pdf
[58] www.cse.iitd.ernet.in/~kkb/DL-1.pdf
[59] http://www.cs.man.ac.uk/~horrocks/Slides/IJCARtutorial/Display/
[60] mcs.unife.it/~friguzzi/chapter2.pdf
[61] http://www.cs.man.ac.uk/~rector/papers/common_errors_ekaw_2004.pdf
[62] www.cs.cornell.edu/courses/cs431/2008sp/Lectures/public/lecture-4-09-08.pdf
[63] http://read.pudn.com/downloads77/ebook/293072/Semantic%20Web%20Technologies%20-%20Trends%20and%20Research%20in%20Ontology-based%20Systems(2006).pdf
[64] www.cse.iitd.ernet.in/~kkb/DL-1.pdf
[65] http://www.meteck.org/teaching/OEbook/
[66] *Fondamenti di Intelligenza Artificiale*, Uni. of Bologna, Italy https://www.slideshare.net/StefanoBragaglia/ontology-development

References

[1] K. Amailef and J. Lu. Ontology-supported case-based reasoning approach for intelligent m-government emergency response services. *Decision Support Systems*, 55(1):79 – 97, 2013.

[2] C. E. Areces et al. *Logic Engineering: the case of description and hybrid logics*. PhD thesis.

[3] F. Baader. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] F. Baader, I. Horrocks, and U. Sattler. Description logics. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, pages 135–179. Elsevier, 2007.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

[6] F. Bobillo. *Managing Vagueness in Ontologies*. PhD thesis, 2008.

[7] F. Bobillo and U. Straccia. *fuzzyDL*: An expressive fuzzy description logic reasoner. In *2008 International Conference on Fuzzy Systems (FUZZ-08)*, pages 923–930. IEEE Computer Society, 2008.

[8] K. Bollacker, N. Díaz-Rodríguez, and X. Li. *Extending Knowledge Graphs with Subjective Influence Networks for Personalized Fashion*, pages 203–233. Springer International Publishing, Cham, 2019.

[9] K. Bollacker, N. Díaz Rodríguez, and X. Li. Beyond clothing ontologies: Modeling fashion with subjective influence networks. In V. C. Raykar, B. Klingenberg, H. Xu, R. Singh, and A. Saha, editors, *Machine Learning meets fashion KDD Workshop*, page 1–7. ACM, 2016.

[10] W. N. Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, Institute for Telematica and Information Technology, University of Twente, Enschede, The Netherlands, 1997.

[11] S. Bragaglia, F. Chesani, P. Mello, and D. Sottara. A rule-based implementation of fuzzy tableau reasoning. In M. Dean, J. Hall, A. Rotolo, and S. Tabet, editors, *Semantic Web Rules*, pages 35–49, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[12] A. Carlson, J. Betteridge, B. Kisiel, and B. Settles. Toward an architecture for never-ending language learning. 2010.

[13] C. Carlsson, M. Brunelli, and J. Mezei. Fuzzy ontologies and knowledge mobilisation: Turning amateurs into wine connoisseurs. In *FUZZ-IEEE*, pages 1–7. IEEE, 2010.

[14] L. Chen and C. D. Nugent. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems (IJWIS)*, 5(4):410–430, 2009.

[15] Z. Chen and B. Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[16] J. Davies, R. Studer, and P. Warren. *Semantic Web technologies: trends and research in ontology-based systems*. John Wiley & Sons, 2006.

[17] N. Díaz-Rodríguez. *Semantic and fuzzy modelling of human behaviour recognition in smart spaces. A case study on ambiental assisted living*. PhD thesis, 2016.

[18] N. Díaz-Rodríguez, S. Grönroos, F. Wickström, J. Lilius, H. Eertink, A. Braun, P. Dillen, J. Crowley, and J. Alexandersson. An ontology for wearables data interoperability and ambient assisted living application development. In *Recent Developments and the New Direction in Soft-Computing Foundations and Applications*, pages 559–568. Springer, 2018.

[19] N. Díaz Rodríguez, P. Kankaanpää, M. M. Saleemi, J. Lilius, and I. Porres. Programming biomedical smart space applications with BioImageXD and PythonRules. In *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, SWAT4LS '11, pages 10–11, New York, NY, USA, 2012. ACM.

[20] N. Díaz Rodríguez, O. L. Cadahía, M. P. Cuéllar, J. Lilius, and M. D. Calvo-Flores. Handling real-world context awareness, uncertainty and vagueness in real-time human activity tracking and recognition with a fuzzy ontology-based hybrid method. *Sensors*, 14(10):18131–18171, 2014.

[21] N. Díaz Rodríguez, M. P. Cuéllar, J. Lilius, and M. Delgado Calvo-Flores. A fuzzy ontology for semantic modelling and recognition of human behaviour. *Knowledge-Based Systems*, 66(0):46 – 60, 2014.

[22] N. Díaz Rodríguez, A. Harma, I. Huitzil, F. Bobillo, R. Helaoui, and U. Straccia. Couch potato or gym addict? semantic lifestyle profiling with wearables and fuzzy knowledge graphs. In J. Pujara, D. Chen, B. Dalvi, and T. Rocktäschel, editors, *6th Workshop on Automated Knowledge Base Construction (AKBC) 2017*, page 1–8. NIPS, 2017.

[23] N. Díaz Rodríguez, R. Wikström, J. Lilius, M. P. Cuéllar, and M. Delgado Calvo Flores. Understanding Movement and Interaction: An Ontology for Kinect-Based 3D Depth Sensors. In G. Urzaiz, S. Ochoa, J. Bravo, L. Chen, and J. Oliveira, editors, *Ubiquitous Computing and Ambient Intelligence. Context-Awareness and Context-Driven Interaction*, volume 8276 of *Lecture Notes in Computer Science*, pages 254–261. Springer International Publishing, 2013.

[24] M. d'Aquin and N. F. Noy. Where to publish and find ontologies? A survey of ontology libraries. *Web Semantics: Science, Services and Agents on the World Wide Web*, 11(0):96 – 111, 2012.

[25] R. Evans and E. Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

[26] T. R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[27] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A practical guide to building owl ontologies using the protégé-owl plugin and co-ode tools edition 1.0. *University of Manchester*, 2004.

[28] C.-L. Liu and H.-L. Yang. Applying ontology-based blog to detect information system post-development change requests conflicts. *Information Systems Frontiers*, 14(5):1019–1032, 2012.

[29] N. Matentzoglu, J. Malone, C. Mungall, and R. Stevens. Miro: guidelines for minimum information for the reporting of an ontology. *Journal of Biomedical Semantics*, 9(1):6, Jan 2018.

[30] N. F. Noy, D. L. McGuinness, et al. Ontology development 101: A guide to creating your first ontology.

[31] E. H. Shortliffe. Mycin: Computer-based medical consultations, 1976.

[32] U. Straccia. *Foundations of Fuzzy Logic and Semantic Web Languages*. CRC Studies in Informatics Series. Chapman & Hall, 2013.

[33] U. Straccia. *Foundations of fuzzy logic and semantic web languages*. Chapman and Hall/CRC, 2016.