Introduction to Description Logic(s)

Federico Chesani

May 24th, 2018

Table of contents

Some considerations. . .

- object naturally fall into categories. . .
  - . . . and quite often into multiple categories
- categories can be more general or specific than others. . .
  - . . . this is true for simple as well for complex categories
- objects have parts, sometimes in multiples
- **the relationships among an object's parts is essential to its being considered a member of a category**

Let's focus on *noun phrases*...

...broadly: any syntactic element (as a clause, clitic, pronoun, or zero element) with a noun's function (as the subject of a verb or the object of a verb or preposition) (Merrian-Webster Dictionary)

We would like to represent:

- individuals
- category nouns
- relational nouns

Let's focus on *noun phrases*...

Individuals

E.g., john, david, maria, ...

Category nouns, AKA **Concepts**

Used to describe basic category classes.
E.g., Hunter, Teenager, ...

Relational Nouns, AKA **Roles**

Used to describe objects that are parts or attributes or properties of other objects.
E.g., Child, Age, Born, ...

An ambiguity...

In natural language many nouns can be used to refer as category or as relations. E.g., *child* can be used:

- as a category: a person of a young age
- to indicate a relation: the inverse of parent

In italian, e.g., we can use the term "professore":

- as the category of people teaching;
- as the relation between people in this room and the guy is talking now...

What else do we desire?

- Again, we are really interested in the generalization relation...
  - we would like to define some generalization relation by ourselves...
  - ...but we would like also to automatically *infer* generalization hierarchies as a consequence of the description we have made of concepts.
- we would like to represent complex concepts as the results of some "composition" of simpler concepts
- we would like to know if an individual belongs to some category or not

Description Logic is a (family of) logic that focus on the description of the terms.

DL vs. FOL

- FOL focuses on sentences
- FOL does not help you on reasoning on complex categories.

Example

We can say that X is a hunter by a 1-ary predicate $Hunter(X)$; similarly, we can say $Gatherer(X)$. What if we want to say that X is both a hunter and a gatherer?
We could ad an axiom:
$Hunter\&Gatherer(X) \leftarrow Hunter(X) \wedge Gatherer(X)$.
But we should do this for every concept for every possible relation among them that we want to capture...

DL vs. Frames

- In Frames the generalization relation is all user defined!
- Roles can have multiple fillers, while slots can't (at least in basic Frames systems)
- Frames provide a procedural solution to the creation and instantiation of individuals

A simple logic: $\mathcal{DL}$

Logical Symbols

Two different sets of symbols: *logical symbols* (with a fixed meaning) and *non-logical symbols* (domain-dependent).

Logical Symbols

- punctuation: (,), [,]
- positive integers
- concept-forming operators: **ALL**, **EXISTS**, **FILLS**, **AND**
- connectives: $\sqsubseteq$, $\doteq$, $\rightarrow$

A simple logic: $\mathcal{DL}$

Non-Logical Symbols

Two different sets of symbols: *logical symbols* (with a fixed meaning) and *non-logical symbols* (domain-dependent).

Non-Logical Symbols

- **Atomic concepts**: Person, FatherOfOnlyGirls (camel casing, first letter capital, same as OOP)
- **Roles**: :Height, :Age, :FatherOf (same as concepts, but precede by columns)
- **constants**: john, federicoChesani (camel casing, but starting with uncapitalized letter)

A simple logic: $\mathcal{DL}$

Complex Concepts

Complex Concepts can be created by combining atomic concepts together, using the concept forming operators

- every 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 $d_1 \ldots d_n$ are concepts, then **[AND** $d_1 \ldots d_n$**]** is a concept.

A simple logic: $\mathcal{DL}$

Sentences

What about sentences?

- 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;

A KB in a description logic like $\mathcal{DL}$ is considered to be any collection of sentences of this form.

A simple logic: $\mathcal{DL}$

A KB in a description logic like $\mathcal{DL}$ is considered to be any collection of sentences of this form.

- constants stand for **individuals** in some application domain;
- concepts stand for **classes or categories** of individuals;
- roles stand for **binary relations** over those individuals.

A simple logic: $\mathcal{DL}$

In many research area there is a distinction between:

- A-Box, Assertion Box: a list of facts about individuals
- T-Box, Terminological box: a list of sentences (also called axioms) that describes the concepts.

In the Semantic Web initiative, such distinction is not so stressed. It happens you can have the former, the latter, or more often, a combination of both.

Concept-forming operators

A desired feature in a description logic is to define complex concepts in terms of more simpler concepts. This is achieved by means of the *concept-forming* operators. We just introduced:

- [ALL r d]
- [EXISTS n r]
- [FILLS r c]
- [AND $d_1 \ldots d_n$]

There are many other concept-forming operators...

Concept-forming operators [**ALL** r d]

[ALL r d]

Stands for those individuals that are *r*-related *only* to individuals of class *d*.

[**ALL** :HasChild Male] All the individuals that have zero or more children, but all males;

[**ALL** :HaveStudents Male] All the individuals that have only male students (o no students at all). In this case with term individuals we could intend universities, schools, courses.

[**ALL** :AreInRoom RoomWhereFedericoIsCurrently] All the individuals in this room;

Concept-forming operators [**EXISTS** n r]

[**EXISTS** n r]

It stands for the class of individuals in the domain that are related by relation *r* to *at least n* other individuals.

[**EXISTS** 1 :Child] All the individuals (the class of the individuals) that have at least one child;

[**EXISTS** 2 :HasCar] All the individuals that have at least two cars

[**EXISTS** 6 :HasWheels] All the individuals that have at least six wheels (individual? in which sense? mind the term!)

Concept-forming operators [**FILLS** r c]

[FILLS r c]

Stands for those individuals that are related *r*-related to the individual identified by *c*.

[FILLS :Child francescoChesani] All the individuals that have has child Francesco Chesani;

[FILLS :HasCar aa123bb] All the individuals that have the car with plate aa123bb;

[FILLS :AreAttendingTheCourse thisCourse] All the individuals that are attending this course.

Concept-forming operators [**AND** $d_1 \ldots d_n$]

[**AND** $d_1 \ldots d_n$]

Stands for anything that is described by d_1 and by $\ldots d_n$. Each individual is a member of all the categories $d_1 \ldots d_n$. If we refer to the notion of sets, here we have the idea of *intersection*.

```
[ AND
      Company
      [ EXISTS 7 :Director]
      [ ALL :Manager [ AND
            Woman
            [ FILLS :Degree phD ]
      ] ]
      [ FILLS :MinSalary $24.00/hour]
]
```

Concept-forming operators

Examples

Example 1

The City Council of Bologna supports some families with cash discounts for the children schools. Such families must have at least three children, and at least one of the parents must be unemployed. Try to model such families using the operators ALL, EXISTS, FILLS, AND.

- which atomic concepts?
- do we need to use some data types, besides the individuals?
- which complex concepts?
- which relations/roles?

Concept-forming operators

Examples

Example 1

The City Council of Bologna supports some families with cash discounts for the children schools. Such families must have at least three children, and at least one of the parents must be unemployed. Try to model such families using the operators ALL, EXISTS, FILLS, AND.

Atomic Concepts

- Person ?
- Children / Parent ?
- Employed / Unemployed ?

Relations

- :MemberOf / :HasMember ?
- :ChildOf / :ParentOf ?
- :Employed / :Unemployed ?

Concept-forming operators

Examples

Example 1

The City Council of Bologna supports some families with cash discounts for the children schools. Such families must have at least three children, and at least one of the parents must be unemployed. Try to model such families using the operators ALL, EXISTS, FILLS, AND.

Atomic Concepts

- Person

Relations

- :HasMember
- :HasMemberChild
- :HasMemberParent

Family – a sad solution...

[**AND**
  [**ALL** :HasMember Person]
  [**EXISTS** 3 :HasMember]
  [**ALL** :HasMemberChild Person]
  [**EXISTS** 1 :HasMemberChild]
  [**ALL** :HasMemberParent Person]
  [**EXISTS** 2 :HasMemberParent]
]

Concept-forming operators

Examples

Example 1

The City Council of Bologna supports some families with cash discounts for the children schools. Such families must have at least three children, and at least one of the parents must be unemployed. Try to model such families using the operators ALL, EXISTS, FILLS, AND.

Atomic Concepts

- Person

Relations

- :HasMember
- :HasMemberChild
- :HasMemberParent

Family – better version...

```
[ AND
  [ ALL :HasMember Person ]
  [ EXISTS 3 :HasMember ]
  [ ALL :HasMemberChild Person ]
  [ EXISTS 1 :HasMemberChild ]
  [ ALL :HasMemberParent [AND
      Person
      EXISTS 1 :HasChild
      ALL :HasChild Person
  ]
```

Concept-forming operators

Examples

Family – better version...

```
[ AND
   [ ALL :HasMember Person ]
   [ EXISTS 3 :HasMember ]
   [ ALL :HasMemberChild Person ]
   [ EXISTS 1 :HasMemberChild ]
   [ ALL :HasMemberParent [AND
       Person
       EXISTS 1 :HasChild
       ALL :HasChild Person
   ]
   [ EXISTS 2 :HasMemberParent ]
]
```

Considerations...

- What about the concept of employed? More troubling, what about unemployed?
- How many parents can have a family?
- Why we do not define the concept of Parent, and then use it inside Family?
- Any relation between the :HasMember and the :HasMemberChild relations?
- Any relation between the parents and the children?

Is it possible that this logic is too poor? What else we would like to have?

Sentences

Sentences are expression that are intended to be true or false in the domain.

$d_1 \sqsubseteq d_2$

Concept d_1 *is subsumed* by concept d_2, i.e. every individual that satisfies d_1 satisfies also d_2

Example: $PhDStudent \sqsubseteq Student$

Every phd student is also a student (but not vice-versa).

Sentences

$d_1 \doteq d_2$

Concept d_1 is *equivalent* to concept d_2, i.e. the individuals that satisfy d_1 are *precisely* those that satisfy d_2

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

A phd student is a student that already graduated, and that has some funding.

Sentences

$c \rightarrow d$

The individual denoted by c satisfies the description expressed by concept d.

Example: *federico* $\rightarrow$ *PostDoc*

Federico is a Post Doc.

Interpretations

An Interpretation $\Im$ is a pair $(\mathcal{D}, \mathcal{I})$ where:

- $\mathcal{D}$ is any set of objects, called *domain*;
- $\mathcal{I}$ is a mapping called the *interpretation mapping*, from the non-logical symbols of $\mathcal{DL}$ to elements and relations over $\mathcal{D}$:
  1. for every constant c, $\mathcal{I}[c] \in \mathcal{D}$;
  2. for every atomic concept a, $\mathcal{I}[a] \subseteq \mathcal{D}$;
  3. for every role r, $\mathcal{I}[r] \subseteq \mathcal{D} \times \mathcal{D}$.

Interpretations

What is the interpretation of complex concepts?

- for the distinguished concept Thing, $\mathcal{I}[Thing] = \mathcal{D}$;
- $\mathcal{I}[[\textbf{ALL } r\ d]] = \{x \in \mathcal{D} \mid \text{for any y, if } \langle x, y\rangle \in \mathcal{I}[r], \text{ then } y \in \mathcal{I}[d]\}$;
- $\mathcal{I}[[\textbf{EXISTS } n\ r]] = \{x \in \mathcal{D} \mid \text{there are at least } n \text{ distinct } y \text{ such that } \langle x, y\rangle \in \mathcal{I}[r]\}$;
- $\mathcal{I}[[\textbf{FILLS } r\ c]] = \{x \in \mathcal{D} \mid \langle x, \mathcal{I}[c]\rangle \in \mathcal{I}[r]]\}$;
- $\mathcal{I}[[\textbf{AND } d_1 \ldots d_n]] = \mathcal{I}[d_1] \bigcap \ldots \bigcap \mathcal{I}[d_n]$

Interpretations and sentences: the basic notion of Entailment

Given an interpretation, when a sentence is true?

Given an interpretation $\Im = (\mathcal{D}, \mathcal{I})$, a sentence α is *true* ($\Im \models \alpha$), according to the following:

1. $\Im \models (c \rightarrow d)$ iff $\mathcal{I}[c] \in \mathcal{I}[d]$;
2. $\Im \models (d \sqsubseteq d')$ iff $\mathcal{I}[d] \subseteq \mathcal{I}[d']$;
3. $\Im \models (d \doteq d')$ iff $\mathcal{I}[d] = \mathcal{I}[d']$;

We use the notation $\Im \models S$, where S is a set of sentences, to mean that all the sentences in S are true in $\Im$.

If there exists an interpretation $\Im$ such that $\Im \models S$, we say that $\Im$ is a model of S.

Entailment

Entailment is defined as in First Order Logic:

A set S of sentences logically entails α if for every interpretation $\Im$, if $\Im \models S$ then $\Im \models \alpha$.

Reasoning on the TBox

In description logic the are some fundamental questions that we would like to get an automatic answer to. Given a knowledge base expressed as a set S of sentences:

1. **Satisfiability**: A concept d is satisfiable with respect to S if there exists an interpretation $\Im$ of S such that $\mathcal{I}[d]$ is nonempty. In such cases, $\Im$ is a model of d.
2. **Subsumption**: A concept d is subsumed by a concept d' w.r.t to S if $\mathcal{I}[d] \subseteq \mathcal{I}[d']$ for *every* model $\Im$ of S.
3. **Equivalence**: Two concepts d and d' are equivalent with respect to S if $\mathcal{I}[d] = \mathcal{I}[d']$for *every* model $\Im$ of S.
4. **Disjointness**: ...

Reduction to Subsumption

Satisfiability, Equivalence, and Disjointness can be computed by reducing them to the only concept of subsumption. The following proposition hold:

Reduction to Subsumption

For concepts d and d':

- d is unsatisfiable $\Leftrightarrow$ d is **subsumed** by $\bot$.
- d and d' are equivalent $\Leftrightarrow$ d is **subsumed** by d' and d' is subsumed by d.
- d and d' are disjoint $\Leftrightarrow$ $d \bigcap d'$ is **subsumed** by $\bot$

Reasoning on the ABox

What about the individuals? We would like to answer also to the question: *does a constant c satisfies concept d?*

With respect to a set S of sentences, this amounts to ask: $S \models (c \rightarrow d)$?

Also such question can be reduced to the concept of *subsumption*...

Which type of reasoning?

Summarizing, two fundamental questions that we would like to get an automatic answer to. Given a knowledge base expressed as a set S of sentences:

1. does a constant c satisfies concept d?
2. is a concept d subsumed by a concept d'?

Answering to these questions amount to compute the entailment for the subsumption problem.

We would like to answer these questions independently by the specific domain or interpretation...

Computing Subsumption

Since all the important questions over a set of sentences S can be reduced to test the subsumption property between two concepts, such task is of the utmost importance, from both the aspects of soundness/completeness, and from computational costs

Two main algorithms family for computing subsumption:

- based on *structural matching*: transform a KB in a "normal form", and then check the existence of corresponding atomic concepts for d, d';
- tableaux-based algorithms

Closed- vs. open-world semantics

Differently by many other formalisms, Description Logics are based on an Open World Assumption.

If a sentence cannot be inferred, then its truthness value is unknown.

Note: somehow this characteristics is linked to the idea of distributed information in the Web.

Closed- vs. open-world semantics – Example

In Prolog (CWA)

```
hasOnlyMaleChildren(X)←
        hasChild(X,Y),
        female(Y), !,
        fail.
hasOnlyMaleChildren(X)←
        hasChild(X,Y).
```

```
hasChild(federico,francesco)
male(francesco).
```

In DLs (OWA)

```
(HasOnlyMaleChildren≐[AND
        [EXISTS 1 :HasChild]
        [ALL :HasChild Male]
])
```

```
(federico→[[:HasChild francesco]])
(francesco → Male)
```

Can we infere that federico is an instance of the class HasOnlyMaleChildren ?

Closed- vs. open-world semantics

OWA Example

(HasOnlyMaleChildren$\doteq$[**AND**
[**EXISTS** 1 :HasChild]
[**ALL** :HasChild Male]
])

(federico$\rightarrow$[[:HasChild francesco]])
(francesco $\rightarrow$ Male)

We do not know how many children has federico... it might be that federico has also a daughter.
The fact is that the knowledge base could be incomplete!!!

The "Oedipus example"

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

Notation in FOL...

hasChild(iokaste, oedipus).
hasChild(oedipus, polyneikes).
hasChild(iokaste, ploneikes).
hasChild(polyneikes,thersandros).
patricide(oedipus).
$\neg$patricide(thersandros).

The "Oedipus example"

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

Notation used in these slides...

iokaste$\rightarrow$[**FILLS** :HasChild oedipus].
oedipus$\rightarrow$[**FILLS** :HasChild polyneikes].
iokaste$\rightarrow$[**FILLS** :HasChild polyniekes].
polyneikes$\rightarrow$[**FILLS** :HasChild thersandros].
oedipus$\rightarrow$Patricide.
thersandros$\rightarrow$ $\neg$Patricide.

The "Oedipus example"

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

We want to know...

...if Iokaste has a child that is a patricide and that itself has a child that is not a patricide...

Iokaste$\rightarrow$[**AND**
[**EXISTS** 1 :HasChild]
[**ALL** :HasChild [**AND**
Patricide
[**AND** [**EXISTS** 1 :HasChild] [**ALL** :HasChild $\neg$Patricide]]]
]

The "Oedipus example"

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

hasChild(iokaste, oedipus).
hasChild(oedipus, polyneikes).
hasChild(iokaste, ploneikes).
hasChild(polyneikes,thersandros).
patricide(oedipus).
¬patricide(thersandros).

Hypothesis 1: the child we are looking for is Oedipus

Indeed, we know Oedipus is a patricide... We know also he has a son, Polyneikes, but we do not know if Polyneikes is a patricide or not...

The "Oedipus example"

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

hasChild(iokaste, oedipus).
hasChild(oedipus, polyneikes).
hasChild(iokaste, ploneikes).
hasChild(polyneikes,thersandros).
patricide(oedipus).
$\neg$patricide(thersandros).

Hypothesis 2: the child we are looking for is Polyneikes...

Indeed, we know Polyneikes has a son, Thersandors, and we know Thersandros is not a patricide.....
...but we do not know if Polyneikes is a patricide or not...

The "Oedipus example"

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

Orginal question:

...if Iokaste has a child that is a patricide and that itself has a child that is not a patricide...

We could infer that ...

...under the OWA, the answer is "the KB does not entail the sentence."

BUT THIS REASONING IS NOT CORRECT!

The "Oedipus example" - Solution

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

hasChild(iokaste, oedipus).
hasChild(oedipus, polyneikes).
hasChild(iokaste, ploneikes).
hasChild(polyneikes,thersandros).
patricide(oedipus).
¬patricide(thersandros).

In all possible models, either Polyneikes is a patricide, or he is not...
All possible models can be split up in two classes: one in which Polyneikes is a patricide, one where he is not a patricide.

The "Oedipus example" - Solution

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

hasChild(iokaste, oedipus).
hasChild(oedipus, polyneikes).
hasChild(iokaste, ploneikes).
hasChild(polyneikes,thersandros).
patricide(oedipus).
¬patricide(thersandros).

- In the first class, the child (we are looking for) is Polyneikes.
- In the second class, the child (we are looking for) is Oedipus.

The "Oedipus example" - Solution

Oedipus killed his father, married his mother Iokaste, and had children with her, among them Polyneikes. Finally, also Polyneikes had children, among them Thersandros.

Thus...

In ALL models Iokaste has a child that is a patricide and that itself has a child that is not a patricide.
The answer to the previous question is YES.

Extending $\mathcal{DL}$

It is possible to extend the presented description logic in several directions:

- by adding concept-forming operators;
- by relating roles, and considering also complex roles;
- by adding *rules*.

Bounds on the number of roles fillers

We have already the operator **EXISTS**. We could similarly add the operator **AT-MOST**, where [**AT-MOST** *n r*] describes individuals related by role *r* to *at most n* individuals.

Example

[**AT-MOST** 1 :Child] denotes all the parents that have only one child.

This extension could be dangerous...

What is the meaning of the following concept:
[**AND** [**EXISTS** 4 *r*] [**AT-MOST** 3 *r*]]
How do we treat such situation?

Sets of individuals

It would be nice to specify that a role can be filled only by individuals belonging to a certain set (without recurring to a concept). We could add the operator **ONE-OF**, where [**ONE-OF** $c_1 c_2 \ldots c_n$] is a concept satisifed only by c_i, used in conjunction with **ALL** would lead to a *restriction* on the individuals that could fill a certain role.

Example

(Beatles $\doteq$[**ALL** :BandMember [**ONE-OF** john paul george ringo]])

Implicitly this means that...

...there is an **AT-MOST** restriction limited to 4 on the role :BandMember.

Qualified Number Restrictions

What if we want to describe, e.g., those parents that have at least 2 male children?
We can add the operator [**EXISTS** n r d], meaning all the individuals that are r-related to at least n individuals that are instances of concept d.

Example

[**EXISTS** 2 :Child Male]

Unfortunately, this simple extension increase the computational complexity of entailment (subsumption) . . .

Other Logic operators

We have completely forgot standard usual operators such as:

- **OR**: what if we want to describe all the young and the elder people, but not the adults?
- $\neg$: what if we want to describe all the people that is not instance of a concept d?

What happens to the computational costs?
What happen to the decidability?

Relating the Roles

What if we want to relate the fillers of certain roles? Suppose you want to say that in a small company the CEO and the owner must be the same individual...
We can add the operator [**SAME-AS** r_1 r_2], which equates fillers of roles r_1 and r_2.

Example

[**SAME-AS** :CEO :Owner]

Unfortunately, also this simple extension increase the computational complexity of entailment, and if allowed with *role chains*, can lead to undecidability ...

Complex Roles

Until now we treated roles ad primitive concepts. . . what if, for example, we want to consider a role defined as the *conjunction* of more roles?
And, more interesting, what if we want to add ideas like the *inverse* of a role? Saying for example that :Parent is the inverse of :Child? In particular, this type of information is heavily used within the Semantic Web initiative. . .

Rules

In the language presented here, there is no way of saying, for example, that all instances of one concept are also instances of another.

- we could this by adding definitions of the type $(d_1 \doteq d_2)$, where d_1 and d_2 are complex concepts with their own definition...
- ...but this could lead to several classification problems, when computing subsumption.

Some description logics allows the introduction of rules, like for example:
(**if** d_1 **then** [**FILLS** r c])
Meaning that every individual of class d_1 also has the specified property.

How many logics?

Summing up:

- we started presenting a language, we called it $\mathcal{DL}$;
- we provided its semantics
- we have seen it is nice, but that there could be other nice constructs that could help us...

Where are we going exactly?

An entire family of logics

Description logics is a *family* of logics. Eahc logic is different depending on which operators are admitted in the logic.
Of course, more operators means:

- higher expressivity;
- higher computational costs
- the logic it might result to be undecidable . . .

At `http://www.cs.man.ac.uk/~ezolin/dl/` there is a nice application that shows complexity issues and decidability depending on which operator you decide to include/exclude in your logic. . .

An entire family of logics

A description logic is named upon the operators included.
A minimal, yet useful logic is named $\mathcal{AL}$: Attributive Language. It includes:

- Atomic concept;
- Universal concept (Thing or $\top$)
- Bottom concept (Nothing or $\bot$)
- Atomic negation ($\neg A$), applied only to atomic concepts
- **AND** operator (also called intersection $\sqcap$)
- **ALL** operator (also called value restriction $\forall R.C$)
- [**EXISTS** 1 r] operator (also called limited existential quantification $\exists R.C$)

An entire family of logics

A description logic is named upon the operators included...

- $\mathcal{ALC}$ extends $\mathcal{AL}$ with the negation for concepts ($\mathcal{C}$ stand for Complement);
- $\mathcal{S}$ is synonim of $\mathcal{ALC}$ augmented with *transitive roles*;

An entire family of logics

Following this line, a set of letters indicate the expressivity of the logic and name the logic

- $\mathcal{F}$ Functional Properties;
- $\mathcal{E}$ Full existential Qualification;
- $\mathcal{U}$ Concept Union;
- $\mathcal{C}$ Complex Concept Negation;
- $\mathcal{S}$ is synonim of $\mathcal{ALC}$ augmented with *transitive roles*;
- $\mathcal{H}$ Role Hierarchy;

An entire family of logics

- $\mathcal{R}$ Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
- $\mathcal{O}$ Nominals
- $\mathcal{I}$ Inverse Properties;
- $\mathcal{N}$ Cardinality Restrictions;
- $\mathcal{Q}$ Qualified Cardinality Restrictions;
- $(\mathcal{D})$ Use of datatype properties, data values or data types.;

Which logic for the Web?

When choosing a knowledge representation formalism, there is always a trade-off between performances and expressivity.
The W3C working group defined the Ontology Web Language (OWL), with a set of operators for representing the knowledge [1].
OWL comes out in three different flavour:

- OWL-Lite, is based on $\mathcal{SHIN}^{(\mathcal{D})}$
- OWL-DL, based on $\mathcal{SHOIN}^{(\mathcal{D})}$
- OWL-Full, highly expressive, can be also high-order logic

[1] (in OWL same things are used with different names, e.g. "classes" instead of "concepts")

Which logic for the Web?

OWL provides concept constructors and axioms (from "Handbook of Knowledge Representation", Chapter 3).

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

Which logic for the Web?

OWL provides concept constructors and axioms (from "Handbook of Knowledge Representation", Chapter 3).

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$^-$

OWL-DL Constructors

- `intersectionOf`: a concept is defined as conjunction (AND) of other concepts. E.g.: Human $\sqcap$ Male.
- `unionOf`: a concept is defined as the disjunction (OR) of other concepts. E.g.: Tall $\sqcup$ Nice.
- `complementOf`: a concept is defined as all the individuals that are not members of a given concept. E.g.: $\neg$Male stands for all the females...
- `oneOf`: a concept is defined as a specified set of individuals. E.g.: TeachersOfThisCourse$\doteq \{$ federico, paola, marco$\}$.

OWL-DL Constructors

- `allValuesFrom`: all the individuals that are in relation r only with individuals of a certain class. E.g.: $\forall$hasChild.Male
- `someValuesFrom`: all the individuals that are in relation r *with at least* one individual of a certain class. E.g.: $\exists$hasChild.Male
- `hasValue`: all the individuals that are in relation r only with a certain individual. E.g.: $\exists$hasChild.$\{$francesco$\}$

OWL-DL Constructors

- `minCardinality`: all the individuals that are in relation *r* with at least *n* individuals. E.g.: all the parents that have at least 4 children ($\geq$ 2 hasChild)
- `maxCardinality`: all the individuals that are in relation *r* with at most *n* individuals. E.g.: all the parents that have at most 4 children ($\leq$ 2 hasChild)
- `inverseOf`: relations have a direction from a *domain* to a *range*. It is common to define also the property from the range to the domain. It is useful to define that a relation is the inverse of another one. E.g.: hasChild inverseOf hasParent. If we have (federico hasChild francesco), then it also holds (francesco hasParent federico)

OWL-DL Axioms

- `FunctionalProperty`: if a relation (property) r is defined *functional*, there can be only one value y for each x in relation r. I.e., there cannot be two *distinct* y_1 and y_2 such that we have $(x\ r\ y_1)$ and $(x\ r\ y_2)$. E.g., consider the relation *isToppingOf*, and consider the Parma ham slice s_1, and two pizzas p_1 and p_2. If we have (s_1 isToppingOf p_1) and (s_1 isToppingOf p_2), then we can conclude two different things:
  - p_1 and p_2 are the same pizza...
  - if I stated previously that $p_1 \neq p_2$, then my KB is inconsistent...

Which tools for the Web?

After OWL has been defined, several tools have been developed, in particular to support OWL-DL. E.g.:

- Protégé ontology editor,supports $\mathcal{SHOIN}^{(\mathcal{D})}$
- Pellet, Racer and FaCT++ are reasoners that supports $\mathcal{SHOIN}^{(\mathcal{D})}$

OWL-DL is not the only choice. For example, the ontology SnoMed is based on $\mathcal{EL}$ with additional role properties.

A Link to start with. . .

`http://dl.kr.org/`