Description Logic

`Another way to represent categories and objects`.

1. Introduction

Summing up, we represent, as humans being, a lot of informations by categories and objects. From the logic point of view, there exist several tools that allow us to deal with this kind of representations.

Semantic networks and **frames** were attempts to handle these mental constructions. However, we end up saying that we need a more **formal language** to describe properly the relationships among categories and objects.

We would like to:

- 1^{st} Represent complex concepts as the result of some **composition** of simpler concepts.
- 2^{nd} Know if an individual/object **belongs** to a category or not.

This is possible by **Description Logics**, designed to make easy to describe definitions and properties of categories, following a formalization of the networks mean.

2. A simple logic: *DL*

The most simple Description Logic is *DL* (Description Logics is a set of logics). *DL* is based on two different sets of symbols:

- Logical symbols, they have a fixed meaning.

  `Definition`
  Logical symbols is a set of:

  1. Punctuation (,), [,].
  2. Positive integers.
  3. Concept-forming operators **ALL**, **EXISTS**, **FILLS**, **AND**.
  4. Connectives $\subseteq$, $=$, $\rightarrow$.

  Each *concept-forming operator* has its own meaning and their semantic is influenced by the properties of the instances of a given category.

- Non-logical symbols, their meanings are domain-dependent.

  `Definition`
  Non-logical symbols is a set of:

  1. Atomic concepts.
  2. Constants.
  3. Roles.

> **Complex concepts** can be created by combining atomic concepts together, using the **concept-forming operators**.

In many research area there is a distinction between:

- **A-Box**, used to describe facts about an entity.
- **T-Box**, used to describe properties of a whole category.

Example

Given a category `Person` and an instance of it `Matteo`, we distinguish A-Box from T-Box in this way:

- `Student(Matteo)` is an Assertion-Box, related only to the individual `Matteo`.
- $\forall x \in \text{Person}, \text{Mortal}(x)$ is a Terminological-Box, related to all the instances of the category `Person`.

As we already said, our goals are: first of all, represent complex concepts and recognize if an object belongs to a category. This is reachable by the usage of *concept-forming operators*.

By these tools, we can formalize more complex concepts using several instances coming from different categories and, at the same time, we could describe the properties of instances already in the category to recognize if an unknown object belongs to it or not (instead of classify it retrieving its properties and compare them within the properties of the possible category).

- 1st **[ALL r d]**.

  `It stands for those individuals that are r-related only to inviduals of class d`. In other words, *[ALL r d]* is a new category composed by those individuals that respect the *role r* and they hold the *domain d*.

  #### Example

  1. **[ALL :HasChild Male]** → Defining a new category made of individuals that have a zero or more children, but all males.
  2. **[ALL :HaveStudents Male]** → Defining a new category composed by invididuals that have only male students.

- 2nd **[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`. In this case, we do not have to respect the *domain d* but we accept only new categories within *at least n* individuals.

  #### Example

  1. **[EXISTS 1 :Child]** → Defining a new category that contains all the individuals that have at least one child.
  2. **[ALL 2 :HasCar]** → Defining a new category composed by invididuals that have at least two cars.

- 3rd **[FILLS r c]**.

  `It stands for those inviduals that are r-related to the individual identified by c`. It is the most specific case; we are defining a complex concept related to a specific instance of a category.

  Example

  1. **[FILLS :HasCar aa123bb]** → Defining a new category made of all the individuals that have the car with plate `aa123bb`.
  2. **[FILLS :AreAttendingTheCourse] thisCourse** → Defining a new category that contains all the individuals that are attending this course.

- 4th **[AND $d_1 \dots d_n$]**.

  `It stands for anything that holds all the domains d at the same time`. If we refer to the notion of sets, this concept-forming operator is like the **intersection** between sets.

  Example

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

  Defining a set of companies that holds all the properties described below.

First Order Logic focuses on sentences. A sentence is an expression that can be either true or false depending from the knowledge base.

We can try to determine the truth of a sentence also by Description Logics, using three different syntax rules:

- $d_1 \subseteq d_2$. Concept d_1 is **subsumed** by concept d_2, if and only if every individual that satisfies d_1 satisfies also d_2.

  Example

  PhDStudent $\subseteq$ *Student* → Every phd student is also a student.

- $d_1 = d_2$. Concept d_1 is **equivalent** to concept d_2, if and only if the inviduals that satisfy d_1 are the same that satisfy d_2.

  Example

  PhDStudent = [AND Student Graduated HasFunding] → A phd student is a gratuated student that has some funding for his reasearch.

- $c \rightarrow d$. The individual denoted by c satisfies the description expressed by concept d.

  Example

  federico → *Professor* → Federico is an instance of category `Professor`.

Above the basic idea, we have to provide a formal semantic of *DL*, which is the real meaning of this logic. The formal description of *DL* is given by the notion of **interpretation**.

`Definition`
An Interpretation I is a pair *(D, I)* where:

- *D* is the set of objects, called **domain**.
- *I* is a mapping function, called **interpretation mapping**, that goes from the non-logical symbols (symbols dependent from the domain taking in account) to elements of the domain *D*:
  1. For every constant c, $I[c] \in D$.
  2. For every atomic concept a, $I[a] \subseteq D$.
  3. For every role r, $I[r] \subseteq D \times D$.

Given the notion of *interpretation*, we can use it to determine if a sentence is true or false following the definition below.

`Definition`
Given an interpretation $I = (D, I)$, a sentence α is true ($I \vDash \alpha$), according to the following rules:

1. $I \vDash (c \rightarrow d)$ iff $I[c] \in I[d]$.
2. $I \vDash (d \subseteq d')$ iff $I[d] \subseteq I[d']$.
3. $I \vDash (d = d')$ iff $I[d] = I[d']$.

`Definition`
Given a *set of sentences* S, if all the sentences in S are true in I, we say that I is a **model** of S, $I \vDash S$.

While in First Order Logic we could ask: the predicate is true or false given the knowledge base, in Description Logic there are four different questions, summed up in:

- **Satisfiability**.
- **Subsumption**.

Definition
A concept d is **subsumed** by a concept d' w.r.t. S if $I[d] \subseteq I[d']$ for every model I of S.

- **Equivalance**.
- **Disjointness**.

Why do we introduce only the definition of Subsumption? Satisfiability, Equivalance and Disjointness can be computed by reducing them to subsumption. The following proposition hold:

Definition
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** $\Leftrightarrow$ $d \cap d'$ is **subsumed** by $\bot$.

3. Closed World Assumption vs. Open World Assumption

The set of Description Logics are based on an **Open World Assumption**: if a sentence cannot be inferred, then its truthness value is **unknown**.

Instead in First Order Logic, we assume a **Closed World Assumption**: the informations not stated inside the database program are simply **false**.

With *OWA* we can formalize different contexts in which the unknown informations can take all the possible values.

Example

Given the following problem:
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 can derive the following knowledge base in *FOL*:

```
hasChild(iokaste, oedipus).
hasChild(oedipus, polyneikes).
hasChild(iokaste, polyneikes). obtained
hasChild(polyneikes, thersandros).

patricide(oedipus).
¬patricide(thersandros).
```

and the same concepts by *DL*:

```
iokaste → [FILLS :HasChild oedipus].
oedipus → [FILLS :HasChild polyneikes].
```

```
iokaste → [FILLS :HasChild polyneikes].
polyneikes → [FILLS :HasChild thersandros].

oedipus → Patricide.
thersandros → ¬Patricide
```

We want to know `if Iokaste has a child that is a patricide and itself has a child that is not patricide`.

`1st Hypothesis`
Indeed, we know `Oedipus` is a patricide, and we know also he has a son, `Polyneikes`, but we don't have enough informations to state if `Polyneikes` is a patricide or not.

`2nd Hypothesis`
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.

Under the *OWA*, the answer is: the knowledge base does not entail the sentence; but this reasoning is not correct! All the possible **models** ($I \vDash S$) can be split up in two classes: one in which `Polyneikes` is a patricide, one where he is not a patricide.

The final answer should be always **YES**: in the first class the answer is <`Iokaste`, `Polyneikes`, `Thersandros`>, instead in the second class the answer is <`Iokaste`, `Oedipus`, `Polyneikes`>. In all the models Iokaste has always a child that is a patricide and itself has a child that is not a patricide.

In this way, both cases answer to the original question. However, the huge problem of this approach is given by the total amount of unknown informations, that in *OWA* might be infinite. *OWA* requires a lot of computational effort then *CWA*.

...