32002: AI (Description Logic) **Spring 2014**

Lecture 23: February 27, 2014

Lecturer: K.R. Chowdhary *: Professor of CS*

23.1 Description Logic

Approaches to knowledge representation are sometimes divided roughly into two categories: *logic-based* formalisms, which evolved out of the intuition that predicate calculus, could be used unambiguously to capture facts about the world; and other, non-logic-based representations. The latter were often developed by building on more *cognitive notions* - for example, network structures and rule-based representations derived from experiments on *recall* from human memory and human execution of tasks like mathematical puzzle solving. Even though such approaches were often developed for specific representational chores, the resulting formalisms were usually expected to serve in general use.

Since first-order predicate logic (FOPL) provides very powerful and general machinery, logic-based approaches were more general-purpose from the very start. In a logic-based approach, the representation language is usually a variant of first-order predicate calculus, and reasoning amounts to verifying logical consequence. In the *non-logical* approaches, often based on the use of graphical interfaces, knowledge is represented by means of some ad hoc data structures, and reasoning is accomplished by similarly ad hoc procedures that manipulate the structures, but lack formal (logic-based) semantics.

The $\mathcal{DL}$ evolved from semantic networks to formalize the network representation, while retaining the emphasis on taxonomic structures as an organizing principle.

A Description Logic models *concepts*, *roles* and *individuals*, and their relationships. The fundamental modeling concept of a $\mathcal{DL}$ is the axiom: "a logical statement relating roles and/or concepts". $\mathcal{DL}$ is a family of formal knowledge representation languages. It is more expressive than propositional logic and has efficient decision properties than first-order predicate logic. It is used in formal reasoning on the concepts of an application domain (known as terminological knowledge). It is used for providing a logical formalism for ontologies and the Semantic Web. Modern ontology languages are based on $\mathcal{DL}$, such as OWL (Ontology Web Language).

A $\mathcal{DL}$ describes the domain in term of the following:

- *Individuals* - are the things in the world that is being described. (For example a house, book, ram, john, rita, etc, all starting with lowercase letters).
- *Classes/Categories/Roles* - are sets of individuals. It is a ako (a kind of) concept. A class is a set of all real or potential things that would be in the class. For example, Hunter, Teenager, etc.
- *Properties/Relations* - are used to describe individuals. It is *ako* Roles or relational nouns, used to describe objects that are parts or attributes or properties of other objects. Examples are: Child, Mother, Age, etc.

23.1.1 Concept and Sentence Structures

Two different sets of symbols: *logical symbols* (with a fixed meaning) and *non-logical symbols* (domain-dependent) are used in the description logic:

Following classes of *Logical symbols* are used in $\mathcal{DL}$:	Punctuation: (,), [,] Positive integers Concept-forming operators: $\forall, \exists$, $FILLs, AND$. Connectives: $\sqsubseteq, \doteq, \rightarrow, \sqcup, \sqcap$.
Non-logical symbols:	Atomic concepts: `Person`, `FatherOfOnlyGirls` (camel casing, first letter capital). Roles: `:Height`, `:Age`, `:FatherOf` (same as concepts, but precede by colons). Constants: `john`, `rajanShaw` (camel casing, but starting with uncapitalized letter.

Concepts: As for the semantics, concepts are given a set-theoretic interpretation: a concept is interpreted as a set of individuals and roles are interpreted as sets of pairs of individuals.

Atomic concepts are thus interpreted as subsets of the interpretation domain, while the semantics of the other constructs is then specified by defining the set of individuals denoted by each construct. For example, the concept $C \sqcap D$ is the set of individuals obtained by intersecting the sets of individuals denoted by C and D, respectively. Similarly, the interpretation of $\forall R.C$ is the set of individuals that are in the relationship R with individuals belonging to the set denoted by the concept C.

A desired feature in a description logic is to define complex concepts in terms of more simpler ones. This is achieved by means of *concept-forming operators*: $\exists, \forall, AND, FILLs$. The *complex concepts* is defined as:

every atomic concept is a concept;
if r is a role and d is a concept, then $\forall r.d$ 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; and
if $d_1 \ldots d_n$ are concepts, then $AND\ d_1, \ldots, d_n$ is a concept.

The $\exists$ stands for the class of individuals in the domain that are related by relation r to at least n other individuals, the following can be created as complex concepts:

$\exists$ 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.

The $FILLs\ r, c$ stands for those individuals that are related (*r-related*) to the individual identified by c. For example, "All the individuals that have the car with plate *RJC12* is represented by $FILLs\ HasCar, RJC12$.

The $\forall\ r.d$ stands for those individuals that are *r-related* only to individuals of class d. For example, $\forall\ BeingInThisRoom.PHDStudents$ represents "All the individuals that are in this room and are students."

Sentences: A KB in a description logic like is collection of sentences of like,

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;

For example,

$PhDStudent \sqsubseteq Student$, i.e., Every Phd student is also a student (not vice-versa).

$d_1 \doteq d_2$, i.e., concept d_1 is equivalent to concept d_2 , i.e. the individuals that satisfy d_1 are precisely those that satisfy d_2.

$PhDStudent \doteq AND(Student,\ Graduated,\ HasFunding)$, i.e., a Phd student is a student that already graduated, and that has some funding.

$c \rightarrow d$, i.e., the individual denoted by c satisfies the description expressed by concept d. For example, $rajan \rightarrow PostDoc$, i.e. "Rajan is a Post Doc."

When compare with FOPL, the latter focuses on sentences, and it does not help you on reasoning on complex categories. For example, we can say that X is a hunter by a 1-ary predicate $Hunter(X)$. Similarly, we can say $Shooter(X)$. What if we want to say is that X is both a hunter and a shooter. In predicate logic, it is

$$Hunter(X) \wedge Shooter(X),$$

where as in $\mathcal{DL}$ it is a 2-ary relation

$$Hunter\&Shooter(X).$$

or

$$AND(Hunter, Shooter).$$

In the $\mathcal{DL}$, intersection of concepts, which is denoted $C \sqcap D$, is used to restrict the set of individuals under consideration to those that belong to both C and D. In the syntax of $\mathcal{DL}$, concept expressions are *variable-free*. In fact, a concept expression denotes the set of all individuals satisfying the properties specified in the expression. Therefore, $C \sqcap D$ can be regarded as the first-order logic sentence, $C(x) \wedge D(x)$, where the variable ranges over all individuals in the interpretation domain and $C(x)$ is true for those individuals that belong to the concept C.

The key characteristic features of $\mathcal{DL}$ reside in the constructs for establishing relationships between concepts. The basic ones are *value restrictions*. For example, a value restriction, written $\forall R.C$, requires that all the individuals that are in the relationship R with the concept being described belong to the concept C.

Example 23.1 *For the network representation in figure 23.1, represent it using $\mathcal{DL}$.*

Let us suppose that `Female`, `Person`, and `Woman` are atomic concepts and that `hasChild` and `hasFemaleRelative` are atomic roles. Using the operators *intersection*, *union* and *complement* of concepts, interpreted as set operations, we can describe the concept of "persons that are not female" and the concept of "individuals that are female or male" by the expressions

The Figure 23.1, represents knowledge concerning persons, parents, children, etc. The structure in the figure is to represent the generality/specificity of the concepts involved. For example the link between Mother and Parent says that "mothers are parents" is sometimes called an "IS-A" relationship. For example, if a person has an age, then a mother has an age, too. This is the typical setting of the so-called (monotonic) inheritance networks.

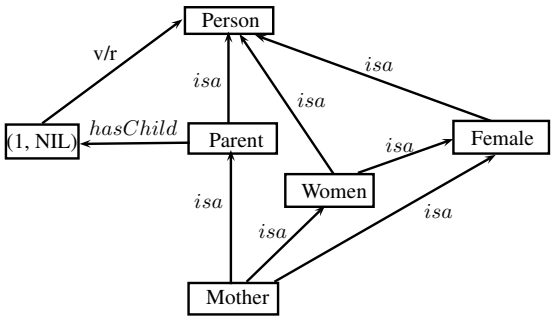

Figure 23.1: Semantic Network Hierarchy.

A characteristic feature of Description Logics is their ability to represent other kinds of relationships that can hold between concepts, beyond IS-A relationships. For example, in Figure 23.1, the concept of `Parent` has a property that is usually called a "role," expressed by a link from the concept to a node for the role labeled `hasChild`. The role has a "value restriction," denoted by the label v/r, which expresses a limitation on the range of types of objects that can fill that role. In addition, the node has a number restriction expressed as $(1, NIL)$, where the first number is a lower bound on the number of children and the second element is the upper bound, and NIL denotes infinity. Overall, the representation of the concept of Parent here can be read as "A parent is a person having at least one child, and all of his/her children are persons."

Relationships of this kind are inherited from concepts to their subconcepts. For example, the concept `Mother`, i.e., a `female parent`, is a more specific descendant of both the concepts `Female` and `Parent`, and as a result inherits from Parent the link to Person through the role `hasChild`; in other words, `Mother` inherits the restriction on its `hasChild` role from Parent.

Observe that there may be implicit relationships between concepts. For example, if we define `Woman` as the concept of a *female person*, it is the case that every `Mother` is a `Woman`. It is the task of the knowledge representation system to find implicit relationships such as these (many are more complex than this one).

Typically, such inferences have been characterized in terms of properties of the network. In this case one might observe that both `Mother` and `Woman` are connected to both `Female` and `Person`, but the path from `Mother` to `Person` includes a node `Parent`, which is more specific then Person, thus enabling us to conclude that `Mother` is more specific than `Person`.

For example, a woman can be defined as a female person by writing this declaration:

$$\texttt{Woman} \doteq \texttt{Person} \sqcap \texttt{Female}$$

However, there are some important common assumptions usually made about $\mathcal{DL}$ terminologies:

- only one definition for a concept name is allowed;
- definitions are acyclic in the sense that concepts are neither defined in terms of themselves nor in terms of other concepts that indirectly refer to them.

The assertion

$$\texttt{Female} \sqcap \texttt{Person}(\texttt{sita})$$

states that the individual `sita` is a female person. Given the above definition of woman, one can derive from this assertion that `sita` is an instance of the concept Woman. Similarly,

$$\texttt{hasChild}(\texttt{sita}, \texttt{luv})$$

specifies that `sita` has `luv` as a child. Assertions of the first kind are also called *concept assertions*, while assertions of the second kind are also called *role assertions*. □

23.1.2 Value Restrictions

Let us now turn our attention to role restrictions by looking first at quantified role restrictions and, subsequently, at what we call "number restrictions." Most languages provide (full) existential quantification and value restriction that allow one to describe, for example, the concept of "individuals having a female child" as $\exists$`hadChild.Female`, and to describe the concept of "individuals all of whose children are female" by the concept expression $\forall$`hasChild.Female`. In order to distinguish the function of each concept in the relationship, the individual object that corresponds to the second argument of the role viewed as a binary predicate is called a role filler. In the above expressions, which describe the properties of tt Parents having female children, individual objects belonging to the concept `Female` are the fillers of the role `hasChild`.

Existential quantification and value restrictions are thus meant to characterize relationships between concepts. In fact, the role link between `Parent` and `Person` in Figure 23.1 can be expressed by the concept expression,

$$\exists\texttt{hasChild.Parent} \sqcap \forall\texttt{hasChild.Person}.$$

Such an expression therefore characterizes the concept of Parent as the set of individuals having at least one filler of the role `hasChild` belonging to the concept Person; moreover, every filler of the role `hasChild` must be a person.

Another important kind of role restriction is given by number restrictions, which restrict the cardinality of the sets of fillers of roles. For instance, the concept

$$(\geq 3\ \texttt{hasChild}) \sqcap (\leq 2\ \texttt{hasFemaleRelative})$$

represents the concept of "individuals having at least three children and at most two female relatives." Number restrictions are sometimes viewed as a distinguishing feature of Description Logics, although one can find some similar constructs in some database modeling languages (notably Entity-Relationship models).

Beyond the constructs to form concept expressions, Description Logics provide constructs for roles, which can, for example, establish role hierarchies. However, the use of role expressions is generally limited to expressing relationships between concepts.

Intersection of roles is an example of a role-forming construct. Intuitively, `hasChild` $\sqcap$ `hasFemaleRelative` yields the role "has-daughter," so that the concept expression

$$\texttt{Woman} \sqcap \leq 2\ (\texttt{hasChild} \sqcap \texttt{hasFemaleRelative})$$

denotes the concept of "a woman having at most 2 daughters".

23.1.3 Reasoning

Given a knowledge base expressed as a set S of sentences:

- "Does a constant c satisfies concept d?"
- "Is a concept d subsumed by a concept $d^{'}$?"

Answering to these questions amount to compute the entailment.

For example, representation for "A Phd student is a student that already graduated, and that has some funding." is:

$$\texttt{PhDStudent} \doteq \texttt{AND(Student, Graduated, HasFunding)}$$

As another example, to say that "Bachelors are unmarried adult males", we write in $\mathcal{DL}$ as $\texttt{Bachelor} \doteq \texttt{Unmarried} \sqcap \texttt{Adult} \sqcap \texttt{Male}$.

The most important aspect of $\mathcal{DL}$ is its emphasis on tractability of inference. A problem instance is solved by designing it and then asking if it is subsumed by one of several possible solution categories.

The complexity of $\mathcal{DL}$ is far simpler than FOPL. The $\mathcal{DL}$ usually also lacks the negation and disjunction operators.

The basic inference on concept expressions in Description Logics is *subsumption*, typically written as $C \sqsubseteq D$. Determining subsumption is the problem of checking whether the concept denoted by D (the *subsumer*) is considered more general than the one denoted by C (the *subsumee*). In other words, subsumption checks whether the first concept always denotes a subset of the set denoted by the second one.

For example, one might be interested in knowing whether $\texttt{Woman} \sqsubseteq \texttt{Mother}$. In order to verify this kind of relationship one has in general to take into account the relationships defined in the terminology in figure 23.1.

23.1.4 Application Domain

1. Software engineering
2. Configuration of large software
3. Digital libraries and Web-based information systems
4. Planning and Data Mining.

References

[1] Chowdhary K.R. (2020) Logic and Reasoning Patterns. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_7