Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

13.1 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.

In a logic-based approach, the representation language is usually a variant of the 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. Among these specialized representations we find semantic networks and frames. However, frames and semantic networks lack formal semantics. Description Logic ($\mathcal{DL}$) was first introduced into Knowledge Representation (KR) systems to overcome these deficiencies of semantic networks and frames. The $\mathcal{DL}$ make it easier to describe definitions and properties of categories. 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, which is more expressive than propositional logic and has more 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).

13.1.1 Definitions and Sentence Structures

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, and used to describe objects that are parts or attributes or properties of other objects. Examples are: Child, Mother, Age, etc.

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

- Punctuation: (,), [,]

- Positive integers

- Concept-forming operators: $\forall, \exists$, *FILLs, AND*.

- Connectives: $\sqsubseteq, \doteq, \rightarrow, \sqcup, \sqcap$.

Non-logical symbols:

- Constants: `john`, `rajanShaw` (camel casing, but starting with uncapitalized letter.

- Atomic concepts: `Person`, `FatherOfOnlyGirls`, `Hunter`, `Teenager` (camel casing, first letter capital).

- Roles: `:Height`, `:Age`, `:FatherOf`, `:Child`, `Mother` (same as concepts, but precede by colons).

13.1.2   Concept Language

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. For example, $Person \sqcap Teacher$. 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. For example, $\forall ResidentsOfJodhpur.Students$

A desired feature in a description logic is to define complex concepts in terms of simpler ones. This is achieved by means of the so-called *concept-forming operators*: $\exists, \forall, AND, FILLs$. A *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 symbol $\exists$ 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."

A knowledge-base in a description logic is collection of sentences 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 is already graduated, and that has some funding.

When compared with FOPL, the FOPL 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

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

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, $C(x) \land D(x)$, 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.

We can represent the concept of "persons that are not female" and the concept of "individuals that are female or male" by the expressions:

$$Person \sqcap \neg Female$$

and

$$Female \sqcap Male.$$

Example 13.1 *Represent the network form of in figure 13.1, 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.

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 13.1, 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 what is called a "value restriction," denoted by the label v/r, which expresses a limitation on the range of

Figure 13.1: Semantic Network Hierarchy.

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 no restriction on the upper limit. 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", thus the role link between `Parent` and `Person` in Figure 13.1 can be expressed as a concept expression $\mathcal{DL}$ as,
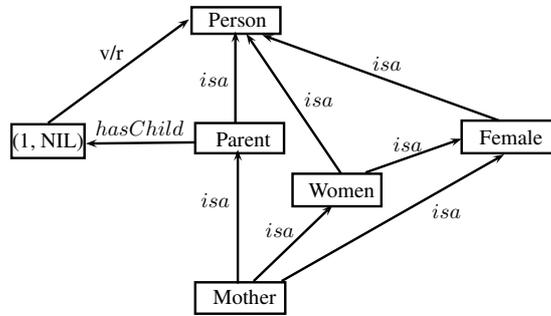
$$\exists 1.\texttt{hasChild}.\texttt{Parent} \sqcap \forall \texttt{hasChild}.Person.$$

Existential quantification and value restrictions are thus meant to characterize relationships between concepts. 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.

Relationships 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`. The other relations are translated as:

$$\texttt{Woman} \doteq \texttt{Person} \sqcap \texttt{Female}.$$
$$\texttt{Mother} \doteq \texttt{Woman} \sqcap \texttt{Female} \sqcap \texttt{Parent}.$$
$$\forall \texttt{Female}.\texttt{Person}$$
$$\forall \texttt{Parent}.\texttt{Person}.$$

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`. □

13.1.3 Architecture for $\mathcal{DL}$ knowledge representation

A knowledge representation system based on description logic provides facilities to setup knowledge-base, to reason about about their concepts, and manipulate them. The figure 13.2 shows the related blocks and their interactions for this purpose.

The KB comprises two components, the $TBOX$ (terminology) introduces the terminology, i.e., vocabulary of an application domain; and the $ABOX$ (assertion box) contains assertions about the named individuals in terms of vocabulary.

The vocabulary consists of concepts, which denotes the individuals, and the roles which denotes the binary relationship between the individuals. In addition to these the $\mathcal{DL}$ system allows the users to build complex description of concepts and roles. The $TBOX$ can be used to assign names to complex descriptions. The description language has models theoretic semantics. Consequently, the semantics in $ABOX$ and $TBOX$ are FOPL formulas or its extensions.

Figure 13.2: $\mathcal{DL}$ based Knowledge representation Architecture.

The $\mathcal{DL}$ system, in addition, provides the services for reasoning using using KB, typically, to reason if terminology is satisfiable. The reasoning process checks that the assertions are consistent. With subsumption testing, it is easy to organize the concepts of terminology in hierarchy.

In any application, KR system is embedded into a large environment. The other components interact until system through queries to KB and by modifying it, i.e., by adding or retracting concepts, roles, and assertions.

The basic form of declaration in a *TBox* is a concept definition, that is, the definition of a new concept in terms of other previously defined concepts. For example, a woman can be defined as a female person by writing this declaration:

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

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 *ABox* contains extensional knowledge about the domain of interest, that is, assertions about individuals, usually called membership assertions. For example,

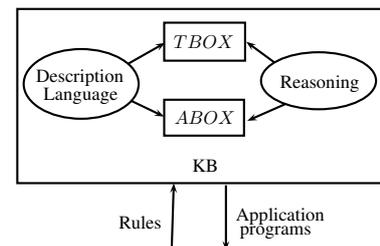$$\texttt{Female} \sqcap \texttt{Person}(\texttt{sita})$$

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,

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

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*.

13.1.4 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 $\exists\text{hasChild}.\text{Female}$, and to describe the concept of "individuals all of whose children are female" by the concept expression $\forall\text{hasChild}.\text{Female}$. 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 `Parents` having female children, individual objects belonging to the concept `Female` are the fillers of the role `hasChild`.

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

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

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).

13.1.5 Reasoning and Inferences

The basic inference on concept expressions in Description Logics is *subsumption*, typically written as $C \sqsubseteq D$ (read as "C is subsumed by D"). Sometimes, this axiom type is also referred to as *is-a* relationship, inspired by the often chosen wording for this type of statement (e.g. "a cat is a mammal" would be a typical verbalization of `Cat` $\sqsubseteq$ `Mammal`). 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.

The principle inferences of $\mathcal{DL}$ are *subsumtion* - checking if one category is a subset of other category, and classification checking, whether an object belong to a category.

For example, one might be interested in knowing whether `Woman` $\sqsubseteq$ `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 13.1.

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

- "Does a constant c satisfies concept d?"

- "Is a concept c 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:

PhDStudent $\doteq$ AND(Student, Graduated, HasFunding).

As another example, to say that "Bachelors are unmarried adult males", we write in $\mathcal{DL}$ as

Bachelor $\doteq$ Unmarried $\sqcap$ Adult $\sqcap$ 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 main application domains of description logic are: software engineering, configuration of large software, digital libraries, Web-based information systems, Planning, and Data Mining.