CHAPTER 6

REASONING WITH EXTENDED DESCRIPTION LOGIC

6.1 INTRODUCTION

The generally accepted framework for the study of reasoning in intelligent systems is a knowledge-based approach. It is assumed that the knowledge is given to the system, stored in some representation language with a well-defined meaning assigned to its sentences. The sentences are stored in a knowledge base, which is combined with a reasoning mechanism, used to determine what can be inferred from the sentences in the knowledge base. Generally, humans handle mundane but complex aspects of the world that occur in everyday situations, in a flexible and speedy manner (Minsky 1981). Any theory aimed at understanding the reasoning process should account for its flexibility, its adaptability and the speed with which it is performed. On the whole, AI community aims at devising reasoning procedures to reflect the human reasoning process. In view of the interdependency of the knowledge representation structure and the reasoning procedures employed, the reasoning algorithms for the extended DL defined in the previous chapter have been explained in this section.

6.2 BASICS OF REASONING

Basically all 'rules of inference' or 'reasoning rules' are the means used to draw conclusions from assertions (Sowa 2000). The conclusions are
dependent upon whether the knowledge stored, is closed world or open world. In closed world knowledge, true propositions can be deduced since any proposition that is not stored or provable is guaranteed to be false. In open world knowledge, since truth-values about some facts are unknown, defaults, probabilities or heuristics have been used to infer missing information (Brachman et al 1985a). These different kinds of reasoning have found applications in diverse fields like economics, linguistics, artificial intelligence and computer science. Researchers in these areas have tended to base their classification and reasoning on philosophical perspectives (Baader and Hollunder 1991). However, pragmatic concerns such as the difficulty of computing and accessing knowledge, which do not find a place in philosophical treatises, have to be addressed (Barr and Edward 1981). Furthermore, handling of negation in reasoning procedures play a significant role in the performance of the inferencing techniques. Several philosophers have dealt with negation from diverse points of view. Plato holds, that ultimately negation, that is, ‘not being’ is the extreme opposite of ‘being’. F.H Bradley accepts that negation is real. Bergson says that negation is subjective since it is relative to the human mind (Bani 2001).

Thus, the close interconnection between the types of knowledge to be used, their representation and the kind of application of the knowledge to be utilized, leads to important conclusions about the reasoning method to be employed. The basic reasoning services of the knowledge representation formalism DL are satisfiability checking, consistency checking, instance checking and subsumption checking. This chapter deals with these reasoning services and the modified reasoning procedures for extended DL.
6.3 TYPES OF REASONING

As mentioned earlier, generally in AI, the basic reasoning classification includes the three major forms of inference deduction, induction and abduction (Edward 2002). When the conclusion of an argument is correctly deducible from its premises, the inference from the premises to the conclusion is said to be deductively valid, irrespective of whether the premises are true or false. But the province of logic, as an exact science does not include all types of deductive reasoning. Thus, it is to be said that logic is the systematic study of the structure of propositions and of the general conditions of valid inference by a method, which abstracts from the content of the propositions and deals only with their logical form. Based on this, and the available knowledge, the advanced classification of reasoning includes non-monotonic reasoning, default reasoning, approximate, probabilistic, minimalistic reasoning and many more. However, in reasoning process relations and negations have a crucial role. The following sections discuss the role of them in detail.

6.4 RELATIONS AND REASONING

In general, knowledge representation scheme needs to associate relations between concepts defined within it (Horrocks and Stephen 2000) (Iwanska and Shapiro 2000). Enhancing the number of relations that can be defined at the highest level of abstraction allows a wider form of inferencing about the entities. The Nyaya theory has propagated the existence of additional relations between concepts at the inferencing level. Some of them are between concepts and some of them are between concepts and qualities at the definitional level. The relations can be between concepts or between a concept and a quality as illustrated in the following Figure 6.1.
As already discussed, Nyaya shastra defines many more fundamental relations at the highest level of abstraction. Among them, some relations have been abstracted, and categorized, depending on its occurrence. The following list specifies a few among them:

**Types of relations only between concepts**

- Part-whole, Generality, Contact-action, Contact-contact, Pervade, Use

**Types of relations only between concepts and qualities**

- Inherence

**Types of relations between concepts and between concepts & qualities**

- Absence-temporal, Presence-temporal, Cause-effect, Limit, Determinant, Qualify, Absence-environment

The existence of relations exhibits the constraints between concepts. For example the relation *Inherence* can exist only between a concept and its associated quality where the concept can be at any level in the hierarchy. The
relation *contact-contact* can exist between concepts at various levels in the knowledge hierarchy. These relations between concepts and between concepts and qualities improve the behavior of the reasoning procedures. Once a relation between concepts is established at the zeroth level the concepts at the lower level automatically inheres. For example, the concept *fire* at zeroth level has qualities and values associated with it. One of the quality of the concept *fire*, *touch* has *hot* as its value and the quality *touch* is related to the concept *fire* by the relation *inherence*, which is one among the relations listed above. In the concept hierarchy, there may be many concepts, which are derived from the concepts at the top level. The word ‘fire’ may be derived from the concept *fire*. If any assertion like ‘fire is not hot’ exists, then the system will reason it in a way that ‘since hotness is an inherent property of fire’, this cannot be true.

6.4.1 Axioms based on relations

Thus, the relations represented in the knowledge hierarchy bring in a new perspective to the interpretation of the represented knowledge. The knowledge can be reasoned using the following axioms:

i) If there exists a relation *R_i* between concepts *C_i* and *C_j*, this implies that concept *C_k* becomes existence, while reasoning with the hierarchy.

ii) If a relation *R_i* exists between *C_i* & *C_j* and a relation *R_m* between *C_i* and *C_k* it implies that a relation *R_n* exists between *C_i* and *C_k*.

iii) If a relation *R_i* exists between *C_i* and *C_j* and concept *C_k* is available, it implies that *C_i* is related to *C_k* by relation *R_j*. 
iv) If a relation $R_i$ exists between $C_i$ and $C_j$ and concept $C_k$ is available it implies that relation $R_k$ relates $C_j$ to $C_k$.

v) If relation $R_i$ exists between $C_i$ and $C_j$ and a relation $R_k$ between concepts $C_j$ and $C_k$ it implies that relation $R_m$ relates $C_i$ to $C_k$.

Figure 6.2 Axioms based on relations
These axioms derive additional information from the represented knowledge.

### 6.5 NEGATIONS AND REASONING

Generally the existence or absence of concepts and their relational existence or absence in the knowledge hierarchy brings in a new perspective in the understanding of the represented knowledge (Alexander 1983). The absence can be viewed as 'not available' or negation of entities. Moreover, negation occupies a unique position in the history of human thought and knowledge. Negation is a keystone in Nyaya shastra. According to Nyaya shastra negation can be cognition, defined as that realizes 'existence of existence cannot be possible without the non-existence of that'. The different factors involved in negation allows for various kinds of existence and non-existence of knowledge like Prior non-existence, Posterior non-existence, Absolute non-existence and Mutual non-existence. In this work, the universal or classical negation, which is generally known as NOT, has been considered along with two other forms of negation known as posterior negation (NOT1) and prior negation (NOT2). These operators allow one to express state changes of concepts involved with either single or multiple concepts where the change in state may be reversible or irreversible.

The various relations between concepts, between concepts and its member qualities and different flavors of negation have been incorporated to enhance the knowledge representation. This enhanced knowledge representation calls for modified methodologies for accessing and reasoning with the knowledge.

The following section deals with different flavors of negation and their semantics.
6.5.1 **Negation Operators**

This section explains posterior negation, prior negation and universal negation.

6.5.1.1 **Posterior negation (NOT1 or \(-1\))**

This type of absence or negation has been characterized as beginning less but having an end. This explains that the state of a thing is not known beforehand but from time ‘t’ onwards the existence of negation becomes true (Figure 6.3).

![Figure 6.3 Posterior Negation](image)

Here posterior negation or absence, describes the situation where earlier to ‘t’ C may be or may not be Q, which is not known, but after ‘t’ C is not Q, indicating the validity of existence of negation.

6.5.1.2 **Prior negation (NOT2 or \(\neg2\))**

This has been characterized, as a thing has its beginning but no end. This explains that it is not that till this moment ‘t’ but it is not known afterwards; meaning the absence may or may not be there, which is shown in the following diagram (Figure 6.4).
Here prior negation or absence indicates that C is not Q before 't' but after 't' the negation or absence may or may not be true.

6.5.1.3 Universal negation (NOT or ¬)

This has been characterized as, 'this is not that' and this is the standard or classical negation operator used generally in reasoning. This 'NOT' indicates that always this is not that. This is not limited to any portion of time. Always the concept C is not Q (Figure 6.5).

6.5.2 Using negation operators

The negation of concepts has been widely used in any knowledge representation system to model the domain in a more natural way. Complicated statements can be expressed using these operators, universal negation (NOT) and additional negations NOT1 and NOT2. In order to express these notions the augmentation of negation operators NOT1 and NOT2 to the existing operators
has to be elaborately discussed to show how these operators function while deriving or inferring information from the represented knowledge.

It is vital to understand the 'power' of these two new negation operators. These operators allow one to express reversible and irreversible state changes associated with either single or multiple concepts. These changes are normally associated with the value of mandatory qualities of the concepts. In other terms, the operators enable recording of natural state change phenomena as an inherent component of the concept definition itself. The presence of both NOT1 and NOT2 indicates relation between two concepts, a reversible change, while the presence of either NOT1 or NOT2, but not both indicates an irreversible change.

The following example shows the representation of the knowledge of a reversible state change where NOT1 and NOT2 can be associated. The two concepts water and water-vapour cannot be associated with a NOT operator, since that would mean water can never become water-vapour or vice-versa. However, if a NOT1 or NOT2 operator connects them, it means that though water and water vapour are not the same, at some point water may become water vapour and vice-versa. This behavior can be expressed in other knowledge representation schemes only with the help of a large number of externally induced relationships, which are not part of the inherent definition of the concept.

The representation of irreversible state change can be recorded using either NOT1 or NOT2 operator. In chemical domain, sulphur changes its crystalline nature due to some change in its inherently associated quality, but is
still called *sulphur*. This knowledge can be represented in a natural way using the negation operator NOT1 as

\[ \text{sulphur: (NOT1) sulphur} \]

Similarly,

\[ \text{ripe-fruit: (NOT1) fruit} \]

indicates that the concept fruit ceases to exist and becomes ripe-fruit after some time ‘t’.

### 6.5.3 Negation handling procedure

These two operators can be used with other operators to represent the knowledge using any knowledge representation language. The behavior of these operators is similar to that of the universal negation except when they concatenate with universal negation. For example, if a concept is defined as follows:

\[ D = (\text{NOT } C) \text{ and } E = (\text{NOT1 } D) \text{ then it derives the following} \]
\[ E = (\text{NOT1 } (\text{NOT } C)). \]

The reasoning procedure to find out whether C subsumes E or not needs a methodology for reasoning to find out the falsity or truth of concatenation of these operators. The following expressions provide the values to decide on the validity of the expression especially when finding the subsumption relationships between the concepts:
NOT and NOT1 --- NOT
NOT and NOT2 --- NOT
NOT1 and NOT --- NOT
NOT2 and NOT --- NOT
TRUE and NOT1 --- NOT1
TRUE and NOT2 --- NOT2
NOT1 and TRUE --- NOT1
NOT2 and TRUE --- NOT2
NOT and TRUE --- NOT
NOT1 and NOT2 --- NOT1
NOT2 and NOT1 --- NOT

The deployment of this, in reasoning procedures which take care of deriving inferred knowledge from the available information, has to be discussed. The following section describes the reasoning services of standard DL.

6.6 REASONING SERVICES OF STANDARD DL

Reasoning mechanisms to infer additional information from the represented knowledge are essential for a knowledge-based system using logics. A characteristic feature of DL, is its ability to infer using reasoning services satisfiability checking, consistency checking, subsumption checking, and instance checking. These inferencing services bring out the implicitly available knowledge in the represented information.
Satisfiability checking essentially checks whether some interpretation of a concept satisfies the axioms that make up the model by which the concept is defined. This reasoning mechanism works by checking concepts against their Tbox definitions. Consistency checking involves the checking of the assertions defined in Abox with respect to the model defined in Tbox. Subsumption essentially checks the hierarchical relation between concepts as defined by Tbox. In other words, the subsumption algorithm tries to fit a concept in the hierarchy by checking whether a concept C subsumes concept D or concept C is subsumed by D. Instance checking essentially checks whether assertions are entailed by definitions of Abox.

Among the reasoning services, the algorithm for subsumption checking involves complex procedures. There are two main strategies for calculating subsumption in DLs (Baader and Sattler 2000). The first approach is based on tableaux-like theorem proving techniques. The key idea is to check whether a given formula F is a logical consequence of a given theory T, which tries to build suitable propagation rules of the generic model T where F is false. If it succeeds the answer is NO, since F is not the logical consequence of T. If it fails the answer is YES, since F is indeed a logical consequence of T. The propagation rules come directly from the semantics of constructors. The second approach is the structural one, which starts by normalizing each description into a canonical form where all implicit information is made explicit but redundant information is eliminated. For every constructor in the candidate subsumer normal form checks the corresponding component of the candidate subsumee and compare the corresponding entries to make sure that the subsumee implies the subsumer. For a simple DL, which supports conjunction, universal quantification and existential qualification, the subsumption algorithm based on the structural approach is as follows:
• Convert to normal form
• Test each term recursively

To test whether C subsumes D,
Check each term $C_j$ satisfies

(a) $C_i$ is in the form atomic or there exists then there is a $D_j$ with $D_j = C_i$

(b) $C_i$ is in the form $\forall R.C$ then there is a $D_j$ with $D_j = \forall R.D'$ and $C'$ subsumes $D'$.

The next section deals with the modifications that have been made in the basic reasoning services of DL.

6.7 REASONING SERVICES OF EXTENDED DL

Generally a knowledge representation system based on DLs is able to perform specific kinds of reasoning. The research associated with language extensions has investigated the semantics of the proposed extensions and often emphasis is laid on finding reasoning procedures for the extended language. The reasoning mechanisms associated with representation methodology provide a powerful knowledge representation scheme for the representation of knowledge from domain dependent lower level information to generic upper-level information. The basic reasoning services of the representation formalism are dependent on the operators, which strengthen the expressiveness of the representation language at the cost of increasing the complexity of the reasoning algorithms.
The additional representational features adapted from Nyaya, which have been incorporated in extended DL, ought to be reflected in the reasoning procedures. Reasoning procedures of extended DL, use the quality dominance definition of NCM for appropriately associating qualities with concepts at different levels, and the extension, shared extension and intension definitions of the model, for determining direct, shared and indirect instantiations. As explained earlier, to accomplish the above necessities, NCM defines three distance functions, namely generalization distance, identification distance, and relation distance, which are used to perform various reasoning processes. These measurements are based on the definition of the concept, scope and value of the associated qualities, extension, intension and the existence of concept-quality relation. In addition, NCM defines certain basic metrics and parameters that evaluate the conceptual distance and conceptual similarity between entities at different levels of the Nyaya based ontology. Basically the reasoning services classify the definitions, detect the logically inconsistent definitions, discover implicit subclass relations and check the instances of the concepts. The following section explains the utilization of various metrics of NCM in the reasoning procedures.

6.7.1 Reasoning tasks for Aboxes

Important problems for an Abox are to find out whether its set of assertions is consistent, which indicates that it has a model. In addition, it has to check whether the assertions in the Abox entail that a particular individual is an instance of a concept description. Among the reasoning services, the Abox reasoning services handle the instance checking and consistency checking. These services are concerned with the validity and consistency of the assertions of the knowledge specified.
6.7.1.1 Instance checking

Instance checking is the problem of checking whether a concept assertion is satisfied in every model of the knowledge base. Further, the reasoning procedure of instance checking, finds the concept, given the instance name, which is an Abox reasoning service. It is used to check whether the knowledge base entails that an individual is an instance of a concept. This reasoning service can be considered as the central reasoning task for retrieving information on individuals in the knowledge base.

The modified instance checking procedure has to consider automatic inheritance of all the qualities and its associated values, due to the extensions made in the constructors that provide a natural means for specifying complex expressions. In this modified instance checking procedure, as soon as an instance of a concept has been defined, it automatically inherits all the qualities and its values associated with the concept. It is due to the extensions made in the constructors, which provide means for specifying a complex expression naturally. For example, ‘a’ is an instance of a concept definition ‘A’ where ‘A’ may be a simple concept or a complex concept. As soon as the information is received the reasoning procedure automatically fixes all the qualities and values associated with it. If the quality name or the value of the quality has been changed using the Abox statements for that particular instance, then the algorithm for instance checking for extended DL checks the associated quality names and values in Abox apart from checking only the concept name.

6.7.1.2 Consistency checking

The reasoning procedure consistency is used for verifying whether the information contained in a knowledge base is coherent. The reasoning
procedure, consistency checking of the extended DL, finds the consistency of the concept description by considering all the relations existing between the concepts and between concept and quality. It is the problem of verifying whether the knowledge base has a model depending on the existence of a relationship between concepts and this can be classified as follows:

**RELATION EXISTENCE CHECKING:** The existence of relations between any two concepts has been checked in Relation existence checking. To find the relations between the concepts in the hierarchy, the algorithm simply tests the relations to check whether given two concepts are far away in the hierarchy. It utilizes the *relation distance* from NCM to accomplish this job.

**INHERENCE CHECKING:** In this procedure, there is a need to check relations between concept and quality. Among the qualities of a concept, various relations can relate more than one quality to the concept. This particular *inherence* relation checking finds the qualities of the concept, which are associated by this relation to the concept.

**TEMPORALLY CONSTRAINED EXISTENCE CHECKING:** This checking lists all the concepts that are related by the negation operators NOT1 and NOT2 in the hierarchy. This is a special kind of relation checking which is unique to the representation of knowledge based on the extended DL.

### 6.7.2 Reasoning tasks for Tboxes

Typical reasoning services for the terminological boxes are satisfiability checking and subsumption checking. Satisfiability determines
whether a description is satisfiable or not, which is non-contradictory, whereas subsumption checking determines, whether one description is more general than the other, that is, whether the first subsumes the second. The modified algorithm for these two reasoning services handles the richer representation framework based on NCM.

6.7.2.1 Satisfiability checking

Checking the satisfiability of concepts is a key inference. A concept C is satisfiable with respect to a Tbox T if there exists a model I of T such that the concept C denotes a nonempty subset in that interpretation I and I is denoted as a model of C. The modified algorithm for satisfiability, checks for the quality and also the associated values of the quality of the complex concept. In addition, it checks for the negation operators NOT, NOT1 and NOT2. These negation operators play a crucial role in checking the satisfiability of a concept, where the concept may be simple or concept. If an interpretation for the concept definition exists then the ‘satisfiability checking’ returns ‘true’ indicating that there is a model for it. However, the modified algorithm checks for the concepts, qualities, values and negations rather than checking only the concept name as in the satisfiability-checking algorithm.

6.7.2.2 Subsumption checking

The subsumption procedure checks whether a concept C subsumes a concept D or not, by considering all the constructors the language supports. The modified subsumption algorithm has been designed for checking the mandatory and optional qualities and its associated values for finding the subsumption
between two given concepts in the hierarchy. Generally the subsumption algorithm checks the concept names to find whether a concept C subsumes concept D. With subsumption tests, the concepts have been organized into a hierarchy according to their generality.

The modified subsumption algorithm, checks for the concept name. If it is satisfied, then checks are made for the mandatory and optional qualities that characterize each concept. Once the mandatory qualities match, the optional qualities are checked. If the optional qualities match return 'yes'; otherwise if the mandatory qualities do not match, the optional quality list is checked for the missing qualities of mandatory quality list. This algorithm then checks for the negation operators. From the hierarchy the algorithm finds the concatenation of different flavors of negation, which is explained in the following procedure.

Modified subsumption algorithm for the extended Description Logic

{  
If C & D are valid concepts in the hierarchy
SUBSUME (C, D) // procedure subsumption
Compare Dname, Cname;
On match,
   Call procedure DECIDE-SUBSUME;
On no match,
   {
      call procedure QUAL-SUBSUME; break;
      if D has parents, for every parent Pi of D
         Repeat

Store the status of parent Pi;
SUBSUME(C, Pi)

Check the status
if (subsume (or) match (or)
no match & no information about the status))
continue;
if does not subsume
break;

QUAL-SUBSUME (C, D) {Procedure QUAL-SUBSUME}
{
Check the mandatory qualities of C & D
If they match,
{
Check the optional qualities C & D;
If matches return “yes”;
}
else
{
check for the qualities in the optional list
If available
{
return “yes”;
}
else
Return “no”;
}
DECIDE-SUBSUME ()  {Procedure DECIDE-SUBSUME}
{
    Repeat
        Get the status of negation operators;
        Decide subsumption as
        {
            NOT1, if status is P
            NOT2, if status is Q
            NOT, if status is true
            Subsumes, if status is false
        }
    decide subsumption based on the following formula:
    {
        NOT, NOT1: NOT
        NOT, NOT2: NOT
        NOT1, NOT: NOT
        NOT2, NOT: NOT
        TRUE Parent, NOT1: NOT1
        TRUE Parent, NOT2: NOT2
        NOT1, TRUE Parent: NOT1
        NOT2, TRUE Parent: NOT2
        NOT, TRUE Parent: NOT
        TRUE Parent, NOT: NOT
        NOT1, NOT2: NOT1
        NOT2, NOT1: NOT
    }
This subsumption algorithm finds whether a C concept subsumes a concept D by considering the associated qualities, its values and also the operators used while forming the concept expression. The negation operators NOT1 and NOT2 used in the concept expression to define a concept in the hierarchy have been given special consideration as explained in the procedure Decide-subsume to find the existence of subsumption between concepts.

Thus, the reasoning procedures for the extended DL incorporate the required inferencing capabilities of a DL system. A knowledge representation system based on extended DL for the representation of information at various levels has been implemented. The system has been named as KRIL, which is explained in the next chapter. It describes the various features of the developed system such as the definition and manipulation language and reasoning procedures. In addition, the chapter explains an evaluation methodology devised for the performance evaluation of the developed system intended for the contents derived.