Description Logic: A Knowledge Representation Language

Vandana Mohan Patil
Assistant Professor
Department of Information Technology,
R.C.Patel Institute Of Technology,
Shirpur, Maharashtra, INDIA.

ABSTRACT

Description logic (DL) is a language for knowledge representation which is used to represent the terminological knowledge of an application domain in a structured and well-understood way. Two important features of DL are expressivity and decidability. Description Logics use two types of data structures for representing knowledge viz T-Box and A-Box. T-Box stores the basic terminologies of the application domain whereas A-Box consists of the assertions resulted from inference. Description Logics use two types of inference patterns: classification of individuals and classification of concepts. The process of inference is called as Reasoning. There are several types of reasoning. The reasoning procedures used in DL are Decision Procedures.
\termsDescription Logics, knowledge representation language, reasoning \keywordsDescription Logics, knowledge representation language, reasoning, decision procedures.
■

1. INTRODUCTION

DL is a formal language for knowledge representation and reasoning about it. It improves structure of knowledge and can be exploited for driving inference. Important characteristics of Description Logics are high expressivity together with decidability, which guarantee that reasoning algorithms always terminate with the correct answers[1]. As well as, the trade-off between expressive power and inference power is achieved using DLs. DLs are descended from SINs(Structured Inheritance Networks). Brachman had put 3 ideas on SINs which further lead to development of DLs [1,2]: 1. The basic syntactic building blocks are atomic concepts (unary predicates), atomic roles (binary predicates) and individuals (constants). 2. The expressivity of language is restricted because of use of small sets of constructors for building complex concepts and roles. 3. Implicit knowledge about concepts and individual can be inferred automatically with the help of inference procedures. Subsumption relationship between concepts, instance relationship between individuals and concepts are inferred from the definition of the concepts and properties of individuals.
DL is now days used in many applications of intelligent information processing systems. Basically DL uses two types of inference patterns [3]:

(1) Classification of individuals

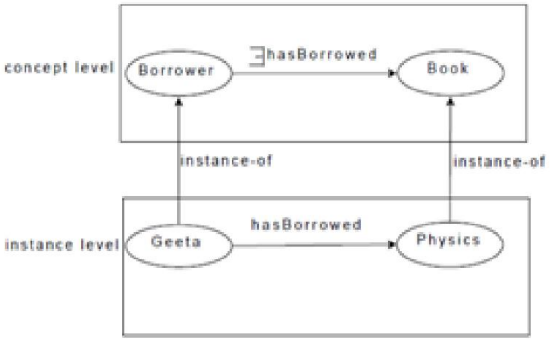

Fig. 1. Classification

(2) Classification of concept

Classification of individuals It determines whether a given individual or object is an instance of a certain concept. It provides useful information regarding properties of individuals. The instance relationship may trigger the application of rules that insert additional facts in knowledge base.

Classification of concept This inference pattern determines the subconcept-superconcept relationships between the concept of given terminology. These relationships are called subsumption relationship in DL. Using this, the terminology can be structured in the form of subsumption hierarchy. The subsumption hierarchy gives useful information which speeds up the inference services. The key characteristic features of DLs are value restrictions which are used for establishing relationships between concepts [4]. The infiniteness of the domain and open-world assumption are distinguishing features of DLs with respect to other modeling languages. The reasoning procedures used in DL are decision procedures. The decision procedures are the procedures which should always terminate both for positive and negative answers.

The two important issues in DL are complexity and decidability. Both depend on expressivity of DL. High expressivity does not

guarantee for inference problems with high complexity or even be undecidable. If DL is less expressive to have high complexity and decidability, it may not be sufficient to express or represent the important concepts of given application. Thats why achieving tradeoff between expressivity and complexity is the most important issue in DL[5].

2. SYNTAX AND SEMANTICS IN DESCRIPTIC LOGIC

This section gives the brief information regarding the syntax and semantics of the Description Logic. The syntax refers to rules for writing sentences in DL whereas semantics state that how these statements will be analyzed[6,7].

2.1 Syntax of Description Logic

The syntax of DL consists of - 1. A set of unary predicates symbols which denote concept names. 2. A set of binary relation or predicates which denote role names. 3. A recursive definition for defining concept terms from concept name and role names using constructors. In DL, concept names are regarded as atomic concept or primitive concept and role names are regarded as atomic roles or primitive role. A concept denotes the set of individuals which belongs to it and role denotes relationship between concepts. The syntax of a member of a DL family is characterized by its recursive definition in which the constructors that can be used to form concept terms are stated. Some common constructors include logical constructors such as intersection or conjunction of concept, union or disjunction of concept, negation or complement of concept, value restriction (universal restriction), existential restriction etc. other constructor may also include restriction on roles such as inverse, transitivity, functionality etc. Description Logic also allows distinction between functional relationships(roles) and non-functional relationships. For example, lentTo is functional while hasBorrowed is non-functional depending upon which concept is filler. The functional roles are sometimes called as attributes or features. There are two types of attributes: partial and total. Total attribute is a attribute for which always there exist some value, for example, a book has a total attribute called ISBN-NO. On the other hand, a attribute is partial if it has certain value at some instant but may not have value at some instant, for example lentTo.

2.2 Semantics of Description Logic

The semantics of DL is defined by interpreting concepts as set of individuals and roles as sets of pairs of individuals. The semantics of non-atomic concept and roles are defined in terms of atomic concepts and roles using recursive definitions.Semantics is given by means of interpretation I that consists of a non-empty set Δ^I (the domain of interpretation) and an interpretation function($.^I$) which assigns to every atomic concept A a set $A^I \subseteq \Delta^I$ and every atomic role R a binary relation $R^I \subseteq \Delta^I \times \Delta^I$ i.e.I=(Δ^I,$.^I$)
The interpretation function is further extended to concept descriptions by the following inductive functions:

$$\top^I = \Delta^I$$
$$\bot^I = \emptyset$$
$$(\neg A)^I = \Delta^I \backslash A^I$$
$$(C \sqcap D)^I = C^I \cap D^I$$
$$(\forall R.C)^I = \{a \in \Delta^I | \forall b.(a,b) \in R^I \rightarrow b \in C^I\}$$
$$(\exists R.\top)^I = \{a \in \Delta^I | \exists b.(a,b) \in R^I\}$$

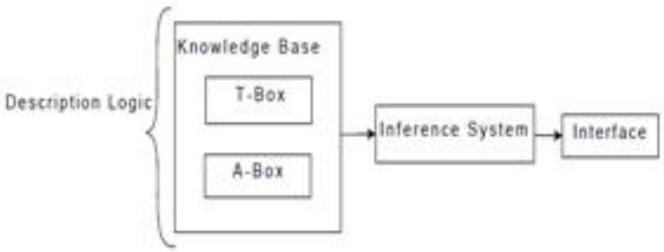

Fig. 2. DL Architecture

3. NOTATIONS IN DESCRIPTION LOGIC [1,2]

The letters A, B are used for atomic concepts, letters C, D are used for concept description, letters R, S are used for roles and letters f, g are used for functional roles (features, attributes). Non-negative integers are denoted by letters n, m and individuals are denoted by letters a, b. This convention is followed when defining syntax and semantics and in abstract examples. In concrete examples following conventions are used:

—Concept names start with an uppercase letters followed by lowercase letters e.g. Human,Male.

—Role names start with lowercase letters e.g. hasChild,marriedTo.

—Individual names are all uppercase e.g. CHARLES, MARY.

Table 3.1:Notations In DL

Constructor	Syntax	Semantics
concept name	A	$A^I \subseteq \Delta^I$
top	$\top$	Δ^I
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^I \cap D^I$
disjunction	$C \sqcup D$	$C^I \cup D^I$
negation	$\neg C$	$\Delta^I \backslash C^I$
value restriction	$\forall$ R.C	$\{a \in \Delta^I \mid \forall b.(a,b) \in R^I \rightarrow b \in C^I\}$
existential	$\exists$R.C	$\{a \in \Delta^I \mid \exists b.(a,b) \in R^I \bigwedge b \in C^I\}$
Unqualified cardinality	$\geq$n.R $\leq$n.R =n.R	$\{a \in \Delta^I \mid\mid \{b \in \Delta^I \mid (a,b) \in R^I\} \mid \geq n\}$ $\{a \in \Delta^I \mid\mid \{b \in \Delta^I \mid (a,b) \in R^I\} \mid \leq n\}$ $\{a \in \Delta^I \mid\mid \{b \in \Delta^I \mid (a,b) \in R^I\} \mid = n\}$
qualified cardinality	$\geq$n R.C $\leq$n R.C =n R.C	$\{a \in \Delta^I \mid\mid \{b \in \Delta^I \mid (a,b) \in R^I \bigwedge b \in C^I\} \mid \geq n\}$ $\{a \in \Delta^I \mid\mid \{b \in \Delta^I \mid (a,b) \in R^I \bigwedge b \in C^I\} \mid \leq n\}$ $\{a \in \Delta^I \mid\mid \{b \in \Delta^I \mid (a,b) \in R^I \bigwedge b \in C^I\} \mid = n\}$

4. THE DESCRIPTION LOGIC ARCHITECTURE[1,8]

As shown in figure,a DL knowledge base consists of two components:

—T-Box : A set of terminological axioms.

—A-Box : A set of assertional axioms

4.1 T-Box

The basic form of declaration in a T-Box is a **concept definition or terminological axioms**. The concept definition is a definition of a new concept in terms of other previously defined concepts or primitive concepts. In other words an equality whose left-hand side is an atomic concept(or role) is called as concept definition.
Terminologies The terminological axioms are used to make statements about how concepts or roles are related to each other. The terminological axioms have the form-

$C \sqsubseteq D$ or $R \sqsubseteq S$

This type of axioms are called as *inclusion axioms* and are useful to state necessary conditions for concept membership using an inclusion when a concept can't be defined completely.
e.g. if one thinks that the definition of woman in the following example is insufficient :
Woman $\equiv$ Person$\sqcap$Female
Man$\equiv$Person$\sqcap\neg$Woman
Mother $\equiv$ Woman $\sqcap\exists$ hasChild.Person
Father $\equiv$ Man $\sqcap\exists$ hasChild.Person
Parent $\equiv$Mother$\sqcup$Father

And also he can't define the concept 'woman' in more detail,then we can use inclusion axioms to define it. i.e.
Woman $\sqsubseteq$ Person
Such type of inclusions are called as 'Specialization' where left hand side of inclusion is atomic.
Another type of axioms is *equalities*. It is of the form-
$C \equiv D$ or $R \equiv S$

An equality whose left-hand side is an atomic concept is called as *'definition'* or *'concept definition'*.Definitions are used to introduce symbolic names for complex descriptions which can be used as abbreviations in other descriptions.
e.g. if we have following definition-
Parent $\equiv$Mother$\sqcup$Father
then we can use it in other definition as follows:

Grandmother$\equiv$Person$\sqcap\exists$hasChild.Parent

E.g. a woman can be defined as a female person by writing this declaration-

Woman $\equiv$Person $\sqcap$ Female
T-Box is made up of set of such concept definitions given that definitions are unambiguous.The T-Box introduces the vocabulary of an application domain. The vocabulary consists of concepts which denote sets of individuals and roles which denote binary relationship between individuals. In addition to atomic concepts and roles users can build complex description of concepts and roles using concept constructors and role constructors. The T-Box can be used to assign names to complex descriptions.

4.2 A-Box

Set of assertional axioms is called A-Box. It contains assertions about named individuals in terms of vocabulary of application domain i.e. T-Box.
There are two type of assertions:

Concept assertions :
The concept assertions are of the type C(a) which states that a belongs to C.
E.g. employee(John)
Role Assertions The role assertions are given as R(b,c) which states that c is the filler of role R for b.
E.g. hasManager(John,Steve)

e.g. Elizabeth and Charles are Persons. We write this as
Person (Elizabeth), and Person(Charles).
Further the assertion Elizabeth is a female person can be written as
Female$\sqcap$Person(Elizabeth)
From this, it can be easily derived that Elizabeth is an instance of concept **Woman**.

Individuals, like "myCar", may have attributes, like "color", and those attributes have values, like "red". This is represented as-
red is the colorOf attribute of myCar. We write this as colorOf(myCar, red).

4.3 Rules

In some DL systems rules are also used to express knowledge in addition to TBox and ABox.The simple form of rule is -
where C and D are concepts and meaning of the rule is if an individual is proved to be an instance of C, then derive that it is also an instance of D. Such rules are called as trigger rules.

5. CONSTRUCTORS IN DL[9,10]

Constructors are used to build complex description of concepts and roles. Accordingly, there are two types of constructors: concept constructors and role constructors.

5.1 Concept constructors

Concept constructors take concept and /or role description and transform them into more complex concept description. Table 5.1 shows the syntax and semantics of common concept constructors.
The things,like kinds of values that can fill roles, and limit on the number of role fillers, are modeled via concept constructors in DL.
The concept constructors are as follows-

—the
—all
—at-most
—at-least
—same as
—non-overlapping
—test
—counting

(1) *the* specifies that the attribute has one and only one value.
e.g. isbnNr

(2) *all* specifies that all individuals should satisfy the given condition.
e.g. Human$\equiv$ Animal $\sqcap$ all hasParent Human

(3) *at-most* specifies the maximum value allowed for the given attribute.
e.g. Mother1$\equiv$Mother$\sqcap$ at-most 3 hasChild

(4) *at-least* specifies the minimum value allowed for the given attribute.
e.g. Borrower ≡ all hasBorrowed Book ⊓ at-least 1 hasBorrowed.

(5) *same as* specifies that two roles have the same value.
e.g same as principal univBOS

(6) *test* is used to check the valid states of the world.
e.g test(date-after(dueDate issueDate) will invoke the function date-after on the passed attributes and check that the first is after the second or not.

Support for enumeration
We can define enumerated set of value for a attribute of a concept using the constructor *the* and *one-of*.
e.g. Book ⊑ (the section(one-of 'student 'reference))
Then we can define corresponding subconcepts as follows:
ReferenceBook≡Book⊓ (fills section 'reference)
Book1≡Book⊓ (fills section 'section)

5.2 Role constructors

Binary relationships are modeled using roles and attributes. There are a number of special constraints on relationships such as : cardinality constraints state the maximum and minimum number of objects that can be related via a role, domain constraints state the kinds of objects that can be related via a role and inverse relationships between roles.
These things are modeled in DL using the role constructors. Cardinality constraints are modeled using the role constructors such as *the*,*at-most*,*at-least*,*all*. Domain constraints are modeled via the role constructor *the* and *all*. Inverse relationships are modeled using the role constructor *inverse*.
e.g.

—Borrower ≡ (and(all hasBorrowed Book)(at-most 2 hasBorrowed))

—Mother ≡ (and Woman (at-least 1 hasChild))

—Book ≡ (and (the hasTitle String) (all hasAuthor Person))

—lentTo≡ ¬ hasBorrowed

6. APPLICATIONS OF DL[11,12]

1. Conceptual modeling It gives the object centered view of the world. Their ontology includes individual objects and their relationships which are grouped together into classes.Binary relationships are modeled directly in DL using roles and attributes. DLs also support for converse relationships using role constructors. As well as it also support for distinction between functional and non-functional relationships. Subconcept-superconcept relationships, abstraction are well modelled by using DL.
2. Software Engineering In this area a notion of a Software Information System (SIS) is developed. One such system is LASSIE, developed to assist the understanding of ATand Ts Definity software system. In this the source code is treated as data and the relationships among them serve the information maintainers during discovery. The knowledge base is populated automatically from source code.
3. Medical Informatics DL is currently applied in at least five applications in medical informatics terminology, intelligent user interfaces, decision support and semantic indexing, language technology, system integration. In these important issues are size, complexity, connectivity, granularity, expressivity etc. DL is capable of handling all of these in well manner. It provides with structured data as well as good reasoning services.
4. Digital Libraries and web based Information Systems For these applications it is much more beneficial if contents are more understandable and available in a machine processable form. Ontologies play an important role in this. OWL is an DL system which properly serves this purpose.
5. Natural Language Processing In these types of applications DL is used to encode some syntactic, semantic, and pragmatic elements needed to drive the semantic interpretation and natural language generation processes. Semantic interpretation is the derivation process from the syntactic analysis of an utterance to its logical form. One part of knowledge base constitutes the lexical semantics knowledge, relating words and their syntactic properties to concept structures, while other part describes the contextual and domain knowledge, giving a deep meaning to concepts. DL also provides the basis for computational processing such as representing common meanings in machine translation applications, answering database queries and dialog management.
6. Databases The application of DL and their reasoning facilities to the database management system includes basically expressing the conceptual domain model/ontology of the data source, integrating multiple data sources and expressing and evaluating queries. Ability of DL to express ontologies at a level closer to that of human conceptualization, determining consistency of descriptions and automatically classifying the descriptions is very useful in these type of applications.

6.1 REFERENCES

(1) Baader,Calvanese,McGuinness,Nardi and Patel- Scheneider, The DescriptionLogic Handbook - Theory, Implementation and Applications, Cambridge University Press, 2003.

(2) http://dl.kr.org, Description Logics, website last access 12 Feb 2014

(3) C. Areces: Logic engineering the case of description and hybrid logics. Thesis, ILLC Univ. Amsterdam, 2000.

(4) E.Franconi,G.DeGiacomo,R.M.MacGregor,W.Nutt,C.A.W elty,andF.Sebatiani,editors.Collected Papers from the International Description Logics Workshop(DL'98) CEUR, May 1998. Pages 55-57

(5) http://www.w3.org, Semantic Web. last access 23 Jan 2014

(6) http://www.loria.fr, DL Applications 11 Mar 2014

(7) Anni-Yasmin Turhan, Description Logic Reasoning for Semantic Web Ontologies, ACM, WIMS11, May 25- 27, 2011

(8) Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, and Marco Ruzzi, Using OWL in Data Integration, Semantic Web Information Management, Springer 2010

(9) Thomas Eiter, G. Ianni, Thomas L., Roman S., Well- Founded Semantics for Description Logic Programs in the Semantic Web, ACM Transactions on Computational Logic, Pages 1-36, April 2010

(10) http://www.ontoprise.de/en/home/products/ontobroker/ last access 20 Mar 2012

(11) http://www.w3.org/TR/2008/WD-owl2-profiles- 20081202 la st access 6 Jul 2013

(12) Brickley D. and Guha R. ,Resouce Description Framework(RDF) Schema Specification , W3C, 1999 at http://www.w3.org/TR/1999/PR-rdf-schema-19990303