The construction of a formal model of an information object in digital library

Oleksandr Novytskyi

Institute of Software Systems NAS Ukraine, UKRAINE, Kyiv, 40, Glushkov Ave,, E-mail: alex.googl@gmail.com

Abstract – The article presents a model of a digital library informational object that aims to preserve and provide access to scientific resources. As a formal representation a UML class diagram is used, to identify semantic and logical contradictions in this diagram is proposed to express it by means of description logic.

Key words – digital libraries, information model object, description logic.

I. Introduction

A digital library in the modern scientific system plays an important role as a mean of delivering information to the final consumer. However, in the realities of the modern world the amount of information, that has to be operated within the digital library, is quite large and constantly growing.

For such digital libraries important issues are the search and integration of information. Particularly, the problems of semantic information integration and semantic search are relevant [1]. As it will be shown below, the search efficiency is directly linked to the conclusion in the ontology.

On the other hand, for software developers it is more convenient by projecting to deal with certain formalisms of knowledge representation models and metadata. For developers the most suitable of such formalisms is class diagram UML [2]. UML is the formalism for software design and analysis. Mane application support the design of large-scale software and provide a user-friendly environment for editing, storing, and accessing multiple UML diagrams.

However, for UML are inherent redundancy and contradiction. Besides there are no machine tools to detect such problems [3]. At the same time, for formal representation of knowledge, such as description logic, the mechanisms for verification of non-contradiction and elimination of redundancy do exist. So naturally there is a problem to combine these two methods for solving the problem of knowledge representation models projecting and DL metadata designing by using UML class diagrams where the problem of redundancy and contradiction is solved.

II. Basic concepts of descriptive logic

For knowledge description it is introduced a kind of abstract language, that is similar to other logical formalisms. The main phase of construction is carried out through two disjoint alphabets of symbols used to denote *atomic concepts* defined as unary predicate symbols , and atomic roles defined as binary predicate symbols , the latter are used to express relationships between concepts. A key feature of DL is the presence of constructs for establishing links between concepts. One of such basic features is a *value restrictions*.

As an example, the value *restrictions* written in the form $\forall R.C$ requires that all objects (individuals), that are related to the concept, should be described as concept C. This means, all objects that are in relation to R objects, are described by the concept in question, and they themselves are described by a set C.

The quenstion of semantics is interpreted from theoretic-plural point of view: the concept is explained as a set of individual objects , and roles are interpreted as the set of object pairs. The areas for interpretation can be chosen arbitrarily, and they can be endless. The domain unlimitedness and assumption world openness are the distinctive features of DL concerning to modeling languages developed in databases .

Atomic concepts are thus interpreted as a subset of interpretation scope, and semantics of other structures is specified by defining the set of objects symbolizing each constructor. For example, the concept $C \cap D$ is a set of objects derived from the intersection of the set of objects marked by C and D, respectively. Similarly, the interpretation $\forall R.C$ is the set of objects that are in relation R to the objects belonging to the set denoted as a concept C.

The basic conclusion about the concept expressed in the DL is *categorization* (*subsumption*), usually presented in the form $C \sqsubseteq D$. The definition of categorization is a problem of validation whether a concept marked as D is more general than the concept C.

Another common conclusion about the expressed concept is a feasibility of the concept; it is a problem of checking whether the expressed concept is a designation of an empty concept. In fact, the conception of feasibility is a special case of categorization, with blank generic concept, which means that the concept is not satisfiable. In the work [4] is suggested, there is a relationship between the expressiveness of the presented language and difficulties of judgments about expressions built using this language. The more expressive a language is, the more difficult are the reasoning.

III. Knowledge representation in DL

The knowledge implementation includes two main aspects. The first one is to provide a precise characterization of the knowledge base; it is a description of the knowledge type and its structure, which will be defined in the system, as well as a clear definition of service judgments for the system to answer those questions for which it was projected. The second aspect is to provide a filled development environment, where the user can profit by using various services which can make the interaction with the system more efficient.

In the knowledge base they can see a clear difference between intensional knowledge or general knowledge about the problem area, and existential (subject) knowledge, specific to a particular task. In DL knowledge base consists of two components $TBox$ and $ABox$.

$TBox$ contains intensional knowledge in the form of terminology (hence the term "TBox") built on declarations that describe general characteristics of concepts. Because of typical categorical relationship between the concepts making up the terminology, $TBox$ is usually regarded as an entity

that has a lattice-like structure; this mathematical structure establishes the categorization of relations and has nothing to do with any of implementation.

ABox contains extensional knowledge, the so-called statement (assertional) knowledge base (hence the term *ABox*), knowledge defined for objects of the conceptual field. Intentional knowledge can acquire the constant modifications.

For the *TBox* terminology determination the special operations are introduced. The canonical form for operations in *TBox* is the *definition* of the concept that means defining a new concept in terms of previously defined concepts. In particular, the main task by terminology constructing is ***classification***, which is equivalent to defining a new concept and proper place in the hierarchy of taxonomic concepts. Classification can be achieved by checking the categorical relationship between each concept in the hierarchy and by extension of the new concept. A place for the new concept is determined between the most specific concepts of the given category, and the most general concepts, with which a new concept is in categorical relations.

ABox

ABox contains extensional knowledge about the conceptual scope, that means, statements about objects; such statements are usually referred as statements of membership. The basic judgment objective in *ABox* is a verification of an object specimen, that is checking out whether a given object in the form of the object specimen belongs to this concept.

DL is a subset of first-order logic. In fact, DL ALC corresponds to the first-order logic fragment obtained by restricting the syntax of formulas containing two variables and operating with unary and binary predicates.

The basic concepts of descriptive logic are atomic concepts and atomic roles. The complex descriptions can be inductively constructed from them by using the concepts constructors. Further we will use letters A, B for denoting the atomic concepts, R for marking atomic roles and letters C, D for concepts and little letters $a,b,....$ to denote the objects [5].

Description languages are distinguished by the constructors, which they represent. AL language is minimal language that offers a practical interest; other languages are an AL extension. This language describes concepts using the following syntax:

For atomic concepts and concepts can be confirmed that every atomic concept is a concept, the converse statement is not true.

It should be noted, that in AL, the negation can be applied only to atomic concepts, and only for the upper (universal) concept the existentional role quantification is allowed.

To define the formal semantics of AL concepts they consider the I interpretations, which consist of non-empty set Δ^I (the interpretation domain) and of the interpreting function that determines for each atomic concept A the set $A^I \subseteq \Delta^I$ and for each atomic role R the binary relation $R^I \subseteq \Delta^I \times \Delta^I$. Interpretation function is extended to describe the concepts with the following inductive definitions:

$$M^I = \Delta^I$$
$$\bot^I = \varnothing$$
$$(\neg A)^I = \Delta^I \setminus A^I$$
$$(C \sqcap D)^I = C^I \cap D^I$$
$$(\forall R.C)^I = \left\{a \in \Delta^I \mid (a,b) \in R^I \rightarrow b \in C^I\right\}$$
$$(\exists R.M)^I = \left\{a \in \Delta^I \mid \exists b : (a,b) \in R^I\right\}$$

It is said that the two concepts C, D are equivalent and it is written $C \equiv D$, if $C^I \equiv D^I$ for all interpretations I.

For extension of the AL language expressiveness the following constructors are introduced. Concepts combining (marked as $\mathcal{U}$) and is written as $C \sqcup D$, is interpreted as $(C \sqcup D)^I = C^I \cup D^I$.

The full existentional quantification (marked as $\mathcal{E}$) and is written as $\exists R.C$ is interpreted as $(\exists R.C)^I = \left\{a \in \Delta^I \mid (a,b) \in R^I \sqcap b \in C^I\right\}$.

Number restrictions (marked as $\mathcal{N}$) is written in form $\geq nR$ (limitation from below) and $\leq nR$ (limitation on the top), where n becomes an integers, and is interpreted as

$$(\geq nR)^I = \left\{a \in \Delta^I \middle| \left|\left\{b \mid (a,b) \in R^I\right\}\right| \geq n\right\}$$

and

$$(\leq nR)^I = \left\{a \in \Delta^I \middle| \left|\left\{b \mid (a,b) \in R^I\right\}\right| \leq n\right\}$$

respectively, where under $|\cdot|$ denotes the cardinality of a se.

Negation of arbitrary concepts (marked C) is written as $\neg C$, and is interpreted as $(\neg C)^I = \Delta^I \setminus C^I$. In such way, by introducing of all above mentioned constructors, we receive the dialect $\mathcal{ALUEN}$. It can be shown that the whole $\mathcal{AL}$ language family can be described using only the constructors $\mathcal{U}$, $\mathcal{E}$, $\mathcal{N}$. Further, we will use the designation $\mathcal{ALC}$ for $\mathcal{ALUE}$ and $\mathcal{ALCN}$ for $\mathcal{ALUEN}$.

IV. Terminology

They enter ***terminological axioms***, which make statements about how concepts and roles are connected with each other. Along with terminological axioms they also introduce the concept of ***determining*** as specific axioms and identify ***terminology*** as a set of definitions, where atomic concepts can be presented, as abbreviations or ***names*** for complex concepts. If determinations in the terminology contain cycles, ***the semantics of stationary points*** is introduced to make the determination monosemantic.

In the general case, terminological axioms have the form $C \sqsubseteq D$ ($R \sqsubseteq S$) - ***axioms of inclusion*** or $C \equiv D$ ($R \equiv S$) - ***axioms of equivalence***, where C, D are concepts and R, S - the roles. Further for simplification we will consider axioms that apply only to concepts.

V. Reasoning

Knowledge representation systems based on DL are able to perform certain kinds of thinking. As mentioned earlier,

the purpose of knowledge representation system goes beyond the conception of preservation of definitions and assertions. Knowledge bases consisting of *TBox* and *ABox* have a determined semantics that makes them equivalent to a set of axioms in first-order predicate logic. So, like any other set of axioms, it contains a hidden knowledge that can be explicitly obtained through **reasoning**.

Different types of judgments are realized by means of DL system and are defined as logic **reasoning**. As it will be shown, one of the major problems of **reasoning** is the check of *ABox* sequence, and all other judgments can be reduced to this problem.

VI. Reasoning in UML class diagrams

UML class diagrams are one of the most important components of UML, further it will be demonstrated a solution for the judgment problem about such schemes.

UML (Unified Modeling Language) is de facto the standard formalism for analysis and software development. One of the most important components of UML are class diagrams, that simulate the information about the field of interests in object terms organized into classes and the relations between them. For such systems is a question open: how difficult it is to perform judgments using UML class diagrams in terms of computing complicacy. DL of the dialect *ALCQI* is quite rich for knowledge representation in terms of concepts (classes) and roles (binary relations). It can be seen as a fragment of DL DLR_{ifd}, where the binarity is limited by the relation and the knowledge base is limited only by the final set of statements only about concepts (there are no statements about relation, no identity relations, no statements of functional dependence, as it requires the relation of 3-rty). Let in *ALCQI* we will use C to denote the concepts and P for marking atomic roles. The syntax *ALCQI* we will construct using the following rule:

$$C ::= A \mid \neg C \mid C_1 \sqcap C_2 \mid (\leq nR.C)$$

$$R ::= P \mid P^-$$

where P^- is an inverse role, which semantics is defined as following $\{(b,a) \mid (a,b) \in P^I\}$.

Moreover, we use following abbreviations:

$\bot$ for $A \sqcap \neg A$ where A is any atomic concept.

$\top$ for $\neg \bot$

$C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$

$C_1 \Rightarrow C_2$ for $\neg C_1 \sqcup C_2$

$(\geq nR.C)$ for $\neg(\leq n-1R.C)$

$\exists R.C$ for $(\geq 1R.C)$

$\forall R.C$ for $\neg \exists R.\neg C$

Knowledge base *ALCQI* consists of the final set of inclusion statement in form $C_1 \sqsubseteq C_2$ where C_1 and C_2 are arbitrarily expressed concepts.

Let's note that for the DL, which does not include structures for constructing the regular expressions on roles, there is a big difference between the judgment methods used in the presence of the knowledge bases, and methods used for judgments on concept expressions.

For example, the logic *ALN* allows for simple structural algorithms in order to solve the problem of judgments, in fact without involving the statements, the necessary polynomial time [6]. However, if we consider free statements, the appropriate developed procedures are raised to the completion strategy power.

VII. The judgment on the UML class diagram using *ALCQI*

A. Classes

UML class C is represented by an atomic concept C. Each attribute R of the type T for the class C is represented by the nuclear role R, along with the inclusion statement, which encodes the attribute R data typification for the class C:

$$C \sqsubseteq \forall R.T$$

It should be kept in mind that other classes can also include the attribute R Multiplicity $[i..j]$ for attributes R are formalized as follows:

$$C \sqsubseteq (\geq iR) \sqcap (\leq jR).$$

- When j is equal to $*$, we reject the second conjunct
- When multiplying relation $[0...*]$, the whole statement is omitted
- When multiplicity is not present ($[1...1]$), the statement will have the following form $C \sqsubseteq \exists R.\top \sqcap (\leq 1R)$

B. Associations

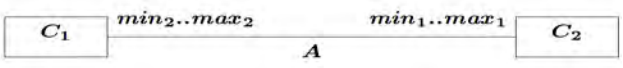

Fig. 1. Associations

Every binary associations (or aggregations) between classes C_1 and C_2 are presented by atomic role A, together with the inclusion statement:

$$\top \sqsubseteq \forall A.C_2 \sqcap \forall A^-.C_1$$

Multiplicativity A is formalized by following statements:

Each exemplar of a concept C_1 is connected through an atomic role A at least with $\min_1$ and no more $\max_1$ exemplars of the concept C_1:

$$C_1 \sqsubseteq (\geq \min_1 A) \sqcap (\leq \max_1 A)$$

Each exemplar of a concept C_2 is connected through an atomic role A at least with $\min_1$ and no more $\max_1$ exemplars of the concept C_2:

$$C_2 \sqsubseteq (\geq \min_2 A^-) \sqcap (\leq \max_2 A^-)$$

It should be noted that aggregation is only a special kind of binary associations without associative class.

C. Generalization

Generalization between the class C and its derived class C_1 can be represented by the inclusion statement $ALCQI \quad C И C_1$. The hierarchy of classes, as shown in Schema 2 can be represented by statements $C_1 И C, ..., C_n И C$. The limitation of classes $C_1, ..., C_n$ that do not intersect, can be modeled as $C_i И у C_j$ де $1 \le i < j \le n$, while the completeness of generalization can be expressed as $C И C_1 g .. g C_n$.

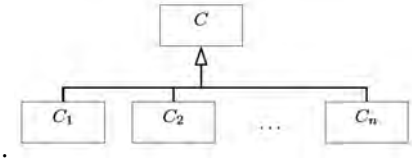

Fig 2. Generalization

Conclusion

For DL an important element for modeling is the representation of data architecture. In the presented UML data model DL allows links between informational resources. The model includes a description of the information resource with metadata, where some of the metadata elements are relations that establish links between informational resources.

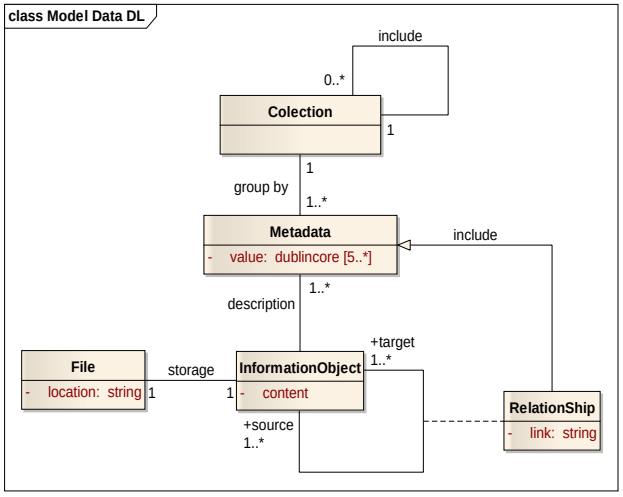

Fig 3. The Model of informational object

Collections are built based on the metadata, in its turn collections can form hierarchical structures. Each piece of information is saved in a file. In the description, metadata must have not less than 5 Dublin Core metadata fields that will provide a minimum data set for informational resources description and for its referring to the collection. Informational objects can be linked with each other through relations that are elements of the metadata. Each informational object has its representation in form of a file.

The formal representation of a given model by means of descriptive logic will be as follows:

$$M И \forall include.Colection h \exists include^{-}.Colection$$
$$Colection И \left(\ge 1 include^{-}\right) h \left(\le 1 include^{-}\right)$$
$$M И \forall groupBy.Metadata h \forall groupBy^{-}.Colection$$
$$Metadata И \forall value.dublincore$$
$$Metadata И \left(\ge 5 value\right)$$
$$InformationObject И \left(\ge 1 target^{-}.RelationShip\right)$$
$$InformationObject И \left(\ge 1 source^{-}.RelationShip\right)$$
$$InformationObject И \forall content.M$$
$$RelationShip И Metadata$$
$$RelationShip И \forall link.String h \forall include.Metadata$$
$$M И \forall storage.InformationObject h \forall storage^{-}.File$$
$$File И \left(\ge 1 storage\right) h \left(\le 1 storage\right)$$
$$InformationObject И \left(\ge 1 storage^{-}\right) h \left(\le 1 storage^{-}\right)$$
$$File И \forall location.String$$

The received formal model can be checked by means of machine output.

For this model we can apply DL reasoning. For example FaCT++ is a DL reasoner designed as a platform for experimenting with tableaux algorithms and optimisation techniques.

References

[1] Rudi Studer (Co-Editor), Paul Warren (Co-Editor) John Davies (Editor), Semantic Web Technologies: Trends and Research in Ontology-based Systems.: Wiley (July 11, 2006), 2006.

[2] Object Management Group, Inc. (2013) Object Management Group - UML. [Online]. http:// www.uml.org/

[3] Diego Calvaneseb and Giuseppe De Giacomoa Daniela Berardia, "Reasoning on UML class diagrams ," Artificial Intelligence, vol. 168, no. 1-2, 2005.

[4] R. J., and Levesque, H. J. Brachman, "The Tractability of Subsumption in Frame-Based Description Languages," in Third National Conference on Artificial Intelligence, 1984, pp. 34-37.

[5] Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider Franz Baader, The Description Logic Handbook.: Cambridge University Press, 2003.

[6] Diego and De Giacomo, Giuseppe and Nardi, Daniele and Lenzerini, Maurizio Calvanese, "Reasoning in expressive description logics," in Handbook of automated reasoning. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 2001, pp. 1581-1634.