Extending a Description Logic to Cope with the Completeness of Multimedia Documents

Carlo Meghini and **Umberto Straccia**[1]

Abstract. Logic is a suitable tool for modelling the retrieval of multimedia documents, as it offers a well founded and universally accepted framework for representing documents and user requests, while viewing retrieval as inference. Especially the latter feature is of extreme interest to information retrieval, as it allows, among other things, to make proper usage of the many kinds of knowledge involved in this task. However, logic as it is also suffers from several drawbacks that make it too rigid to cope with real domains. We address one of these problems, by defining a logic for the retrieval of multimedia documents allowing the expression and proper handling of closure assertions. A closure assertion is meta-knowledge about the completeness of the knowledge on a specified individual or predicate symbol.

1 Introduction

An increasing research effort is being devoted to the main problems posed by the interaction with Multimedia Documents Repositories (MDRs), a major source of information, nowadays. One of the most prominent of these problems is the retrieval of multimedia documents: given an information need, what are the documents that are *relevant* to it?

Traditionally, the retrieval problem has been studied in the Information Retrieval (IR) area, whose main concern has been the development of systems able to provide an effective retrieval service for large text collections. At the root of any such systems, there is the retrieval model, that is a formalism defining in rigourous mathematical terms the representation of documents and queries, as well as which of the former are to be returned in response to which of the latter.

The typical IR system dealing with textual documents is illustrated in Figure 1. A text document is usually represented as pair $d = (\vec{t}, \vec{w})$, where $\vec{t}$ is a vector of tokens appearing in the document and $\vec{w}$ is a vector of weights associated to these tokens. The weight of a token is the result of a complex formula which takes into account the frequency of occurrence of the token in the document, normally relative to the frequency of occurrence in the other documents. Given a query, the system produces a ranked list of the documents that it considers *relevant* to the query. Each document has an associated degree of relevance. Thereafter an iterative interaction between the user and the system takes place, named *relevance feedback,* consisting of the following basic steps:

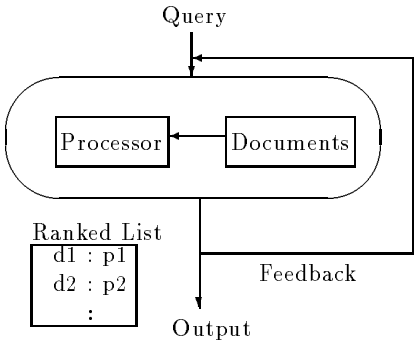

Figure 1. Typical IR system for textual documents.

1. the user selects from the current answer those documents, or parts thereof, wich are deemed as particularly indicative of her information need;
2. the system combines this information with the current query, producing a new, more accurate query;
3. a new answer is produced.

The performance of an IR system is measured in terms of recall and precision: the former counts, in pecentual, how many relevant documents are effectively retrieved, while the latter counts the percentage of relevant documents among the retrieved ones. Recall and precision are defined formally in Table 1 and pictorially illustrated in Figure 2.

Recently, the IR problem has been applied to MDRs, thus opening a new and most exciting research area, *Multimedia Information Retrieval.* When considered in the context of multimedia documents, the problem of retrieval modelling acquires new characters, related to the specific nature of multimedia information. In particular, two features emerge:

- multimedia information does not lend itself to the automatic construction of document representation, at least not

[1] Consiglio Nazionale delle Ricerche, Istituto di Elaborazione della Informazione, Via Santa Maria 46, I-56126 Pisa, Italy

ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

A	=	Relevant Documents
B	=	Retrieved Documents
PRECISION	=	$P(A\|B) = \frac{\|A \cap B\|}{\|B\|}$
RECALL	=	$P(B\|A) = \frac{\|A \cap B\|}{\|A\|}$

Table 1. Definition of precision and recall

as easily as textual information does;

- multimedia information speaks a much richer language than text ("a picture is worth a thousand words").

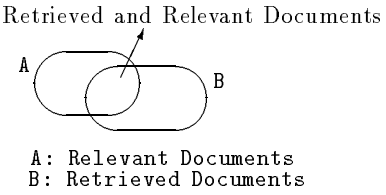

Figure 2. Set representation of PRECISION and RECALL.

While the first feature indicates that automatic document indexing is hardly applicable to multimedia settings, the second calls for sophisticate models, able to account for the audio-visual language of multimedia information.

The founding assumption of our work is that the effectiveness of retrieval in multimedia systems is directly related ot the adequacy of their underlying model. In particular, we believe that the model has to fulfil three basic requirements: (a) be of a logical nature, as question answering is best viewed as logical inference; (b) be powerful enough to capture the many aspects of multimedia information which are not found in traditional information systems, such as data or knowledge bases; and (c) make usage of all sorts of knowledge that may add value to the basic information (i.e. multimedia documents), such as lexical and domain knowledge. In summary, "intelligent" MDRs need Knowledge Representation and Reasoning facilities.

As a consequence, the architecture of a MDR changes significantly. In fact, in a knowledge base based approach to MDRs, a MDR system can be described as an in Figure 3 (note, that our aim is to describe only a possible base schema and not a detailed one).

We will give only an overall description of such a system. There are mainly two important parts in such a system. The *Document Base* (DocB) and the *Document Knowledge Base* (DocKB).

DocB : The DocB is the set of the multimedia documents handled by the system, persistently stored in some way.

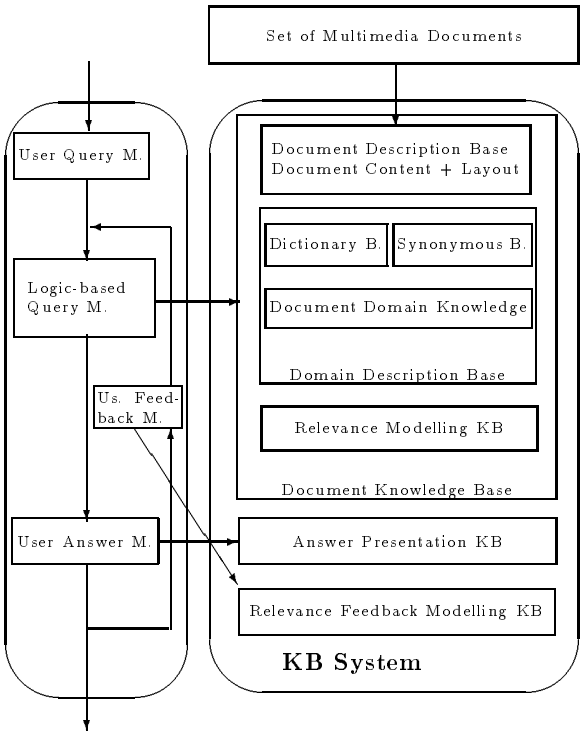

Figure 3. KB based MDR system.

Each document is identified by an unique ID.

DocKB : The DocKB is a set of logical descriptions (formulae of the underlying logic) describing both the documents of the DocB and knowledge about the application domain. The DocKB is partitioned into three subsets: the *Document Description Base* (DocDB), the *Domain Description Base* (DomDB) and the *Relevance Modelling KB* (RMKB).

DocDB : For a document of the DocB, by using indexing technics, pattern recognition technics, natural language processing, manual processing, etc., the system builds logical descriptions about the content and layout of this document. The set of all the descriptions of the documents in DocB forms the DocDB. Therefore, DocDB is the set of logical descriptions of the documents in the DocB.

DomDB : The DomB describes knowledge about the application domain. Typically, it contains three subsets: the *Document Domain Knowledge* (DocDomK), the *Dictionary Base* (DicB) and the *Synonymous Base* (SynB).

DocDomK : The DocDomK contains knowledge about a specific application domain of the system, *i.e.* documents are all about laws, or are invoices, or are about music styles, or are computer science technical reports, or are medical reports, etc. and, thus, includes specific information about document's content and layout.

DicB and SynB : The DicB and SynB contain term and synonymous definitions which are specific to the application domain, as general term definitions.

RMKB : The RMKB models, in logical terms, the notion of relevance, of a document with respect to a query, captured by the system.

The other modules handles the interaction of an user with the system. In particular, the *User Query Module* (UQM) enables an user to formulate a query Q by means of a language, *e.g.* a formal query language as SQL, or natural language, or natural language with images and sound (a multimedia document), or whatever is needed. Note that a query could be also a (short) multimedia document. In fact in this case the goal of the system could be: retrieve those documents whose content is relevant to the content of the query document.

The query Q is successively "formalised" into a formula of the underlying logic. The *Logic-based Query Module* (LQM) is responsible for query solving, and thus queries the DocKB about those documents which are "relevant" to the given query, getting a set of unique document identifiers (the IDs of the relevant documents), called *answer set*. This set is sent to the *User Answer Module* (UAM). This module interacts with the Answer Presentation Modelling KB, which describes what and how the retrieved information are to be presented to the user, *e.g.* the documents themselves, an explanation about the retrieval of each document (*i.e.* why the system has retrieved a document), etc..

Finally, the User Feedback Module (UFM) interacts both with the user and with the Relevance Feedback Modelling KB (RFMKB), which implements a chosen politic in order to handling relevance feedback, enabling to reformulate a new more precise query with respect the user's needs.

Following this line of thinking, we have defined a description logic (DL) [11] for document retrieval and started to investigate what aspects of the problem could not be dealt with properly in the context of the logic.

It should be noted that there are several reasons which makes DLs so attractive for KB-based approach to MDRs (for space limitations, we list only a subset):

- DLs are Object-Oriented Representation Logics: they allow the representation of concepts (classes) and roles (attributes), both structured according to a partial order (inheritance hierarchy);
- DLs have good computational behaviours (see [5, 7]);

and there are a lot of extensions as

Probabilistic extension : Probabilistic versions of DLs [18, 20] could be investigated as a means of making explicit various sources of uncertainty, such as uncertainty related to domain knowledge and uncertainty related to automatic document representation, which is typical in MDRs;

Concrete domain extension : Ability to refer to concrete domain and predicates on these domains [2];

Rule language extension : Rules, as those appearing in the context of frame-based systems (procedural rules), has been shown to be very useful in real applications as they helps to describe knowledge about the domain [9];

Closed World Assumption : Close world reasoning seems to be suitable for IR purposes, as they are close to usual databases reasoning [8, 15];

Temporal/Planing extension : Integrating time into DLs using temporal logics and interval calculus, yielding a temporal DL which combines structural with temporal abstraction [1, 17];

OODBMS extension : Extension about the integration of DLs and Object Oriented Database Systems seems to be very useful for both [4];

Modal extension : Modal operators are integrated into DLs, yielding a modal DL which handles notions as belief, intention and time, which are essential for the representation of multi-agent environments [3];

Default extension : Default inheritance reasoning, a kind of default reasoning that is specifically oriented to reasoning on taxonomies (typical of frame-based systems) is included into DLs [19].

Hence, the present work addressees one such aspects (which is not handled in the above list), namely the adoption of a closed-world reading on certain individuals (document ids) or primitive concepts of the document base.

In essence, we introduce in the assertion language of the logic two special kinds of assertions: individual and predicate closures. By using the former on a certain individual d, the document indexer can limit the specification of d only to the positive information, and have the negative facts (usually overwhelming in quantity) inferred from the document base, which is brought to reason on d in accordance to the closed world assumption [15]. Analogously, by closing a primitive concept A, the indexer will let the derivation of negative ground assertions on A happen as a consequence of the failure of deriving the corresponding positive information.

Both these kinds of closures are essential to multimedia document bases, whose contents are typically objects of a predetermined, rigid structure.

The paper is structured as follows: in the next Section we present the problem solved by closures and in Section 3 we compare our approach with other approaches to CWA. Section 4 formalises closures, whereas in Section 5 we show some properties of our model. Finally, in Section 6 we present a sound and complete proof procedure and its complexity. Section 7 concludes.

2 The problem

The young Adso has been recently hired by the director of a very important library, supported by a sophisticate system for information retrieval (IR), based on logic. Thanks to a recommendation letter from his master Guglielmo, he has been given an important role, that of document indexer, even though at the beginning he will be working on very small and ordinary documents, such as letters. After studying the basics of the system, Adso tackles his first duty, that is to specify the representation of his recommendation letter.

The study of the system has revealed to Adso that letters are instances of the concept `Letter`, so, after selecting the

name d for the one at hand, he enters into the system the assertion:

$$\texttt{Letter(d).}$$

The system promptly replies *Assertion added,* so Adso can proceed to specify the sender of the letter, which he does by inputing the assertion:

$$\texttt{Sender(d,Guglielmo).}$$

Also this time the system reacts in the expected way, and the specification of d goes on until Adso has entered all the available information by telling the system the corresponding assertions, entering Letter(d1) (for which he don't know the sender, yet).

At the end of the specification, Adso wants to check the result of his work by issuing some queries to the system. To begin the checking stage, Adso poses the entered assertions as queries, and the result to all of them is a reassuring *Yes.* Relieved by this success, Adso decides to poses more subtle queries, in order to practice with power of logic. So Adso enters the query:

$$\neg\texttt{Book(d1)}$$

where Book is a document type concept. The expected answer is *Yes,* but, to the surprise of the enquirer, the system answers *I don't know.* Knowing the difficulties of implementing negation Adso moves moves on to universal quantification. After obtaining a *Yes* in response to the query Scottish(Guglielmo), he issues the query:

$$\forall\texttt{Sender.Scottish(d)},$$

asking the system whether all the senders of d are Scottish. As before, a *Yes* answer is expected, because Adso knows that Guglielmo is the only sender of d told to the system. But expectations are again disappointed by another *I don't know* answer. At this point, Adso resorts to a logic textbook and quickly discovers that what was wrong with the previous session were his expectations. In fact, as Adso knows well, the system he is using is based on classical logical implication (in symbols, $\models$) and, from the given premises, it does not logically follow that d is not a letter and that all d's senders are Scottish. Symbolically,

$$\begin{aligned}
\texttt{D} &= \{\texttt{Letter(d)},\texttt{Sender(d,Guglielmo)},\\
&\quad\ \texttt{Scottish(Guglielmo)},\texttt{Letter(d1)}\}\\
\texttt{D} &\not\models \neg\texttt{Book(d1)}\\
\texttt{D} &\not\models \forall\texttt{Sender.Scottish(d)}.
\end{aligned}$$

Having learnt the basics of logical implication, Adso realises that, in order to license the latter inference, the system must be told that there are no other senders of the letter d. This can be done in any DL including number restriction operators by means of the assertion $(\leq\ 1\ \texttt{Sender})\texttt{(d)}$, but, unfortunately the logic underlying the library's system does not include the $\leq$ operator; so Adso has no choice but to add the query itself to the representation of the document. Even in the case of the former inference, the only way out is to tell the system that d1 is not a book. But then, thinks Adso, the same problem would arise with the query:

$$\neg\texttt{Scottish(d)}.$$

At this point, Adso understands that he will obtain the expected behaviour of the system only after telling the whole story about d, that is not only what d is, but also what d is not. After a quick glance to the system catalog, Adso reckons that a complete description of d would amount to a few thousands concept assertions, namely one assertion of the form $\neg A\,\texttt{(d)}$ for all primitive concepts A which d is not an instance of, and a few millions role assertions, namely $\neg R\texttt{(d,}c\texttt{)}$ for all primitive roles R and individual constants c such that c is not an R-filler of d. A comprehensible desperation takes possess of Adso.

In order to overcome this problem, we propose a model which extends the one used by Adso with assertions on individual constants, such as d, having the form:

$$\texttt{Cl(d)}.$$

Informally, an assertion of this kind means that the document base contains, whether explicitly or implicitly, everything that is true about d, and every other piece of information on d is to be understood as false. An assertion like the above one is called a *closure assertion,* as the reading of the information concerning d induced by the assertion is akin to a closed-world assumption. The individuals that are subject to closure assertions are said to be *closed.* Analogously, the assertion Cl(A) "closes" the knowledge on the primitive concept symbol A. Hence, the effect of Cl(Book) is that books are only those *known* to be books in all models of the KB, without bothering to specify ¬Book(a) for all the very many individuals that are not books.

The desired effect of closure assertions is twofold. From one hand, they are meant to give the indexer the possibility of specifying meta-information, regarding the way in which the information on certain individuals is to be considered. From the other hand, they are meant to guide the inferential behaviour of the system on closed individuals and primitive concepts in a way that reflects Adso's, and our own, intuition. More precisely, while the lack of information on non-closed elements is to be interpreted, in the usual way, as evidence of the incompleteness of the representation, the lack of information on closed ones is to be interpreted as evidence to the contrary. Returning to the previous example, this means that the desired interpretation of closure assertions would grant the following inferences:

$$\begin{aligned}
\texttt{D}\cup\{\texttt{Cl(Book)}\} &\models_c \neg\texttt{Book(d1)}\\
\texttt{D}\cup\{\texttt{Cl(d)}\} &\models_c \forall\texttt{Sender.Scottish(d)},
\end{aligned}$$

where $\models_c$ is the inference relation of the new model. $\models_c$ should clearly be non-monotonic, as the addition of knowledge should block the application of closed-world reasoning. For instance, the following should hold:

$$\texttt{D}\cup\{\texttt{Cl(d)}\}\cup\{\texttt{Book(d1)}\} \not\models_c \neg\texttt{Book(d1)}.$$

3 Approaches to CWA in DLs

Since the seminal paper by Reiter [15], many forms of closed-world assumption (CWA) have been investigated (see [10] for a thorough review of this work). Without going into the details of the single proposals, we observe that none of them is able to apply the closure to specified individuals. Furthermore, the results obtained in these studies do not carry over DLs, as they are formulated for universal theories without equality. As it is well known, the first-order counterparts of DLs are calculi with existential quantification (and even equality, if number restrictions are allowed).

For this reason, formulations of the CWA for DLs have appeared which are based on the usage of an epistemic operator [8]. The basic idea behind these proposals is to enforce a CWA reading of the information about an individual `a` by using an epistemic operator **K** in queries regarding `a`. Applied to the previous example, this means that in order to obtain a positive answer on the non-membership of `d1` to the `Book` concept, one has to pose the query:

$$\neg\mathbf{K}\texttt{Book(d1)},$$

asking is whether `d1` *is not known* to the knowledge base to be a book, which is indeed the case. Analogously, the answer to the query:

$$\forall\mathbf{K}\texttt{Sender.Scottish(d)}$$

is *Yes* because there is only one known sender of `d` and he happens to be Scottish.

As made clear by these examples, the usage of an epistemic operator in queries allows one to ask questions not only on how the modelled world is, but also on what the knowledge base knows about such world [14]. It is also evident that this usage permits to capture, among other things, some form of CWA. However, a clear connection between epistemic queries to knowledge bases and any of the various CWA formulations has been established only in a very restricted case (see Theorem 5.1 in [6]); thus, strictly speaking, one cannot claim full control of how epistemic queries realise CWA.

Besides this formal argument, the adoption of the epistemic approach in our setting is problematical due to the fact that queries to document bases are not assertions, but concepts, returning the individuals that, in every interpretation, are instances of the query concept [11]. Now, let us consider the document base `E` and the query `C` defined as follows:

$$\begin{aligned}\texttt{E} &= \{\texttt{Letter(d)},\texttt{Cl(d)},\texttt{Letter(a)}\}\\ \texttt{C} &= \neg\texttt{Book}.\end{aligned}$$

According to our intended meaning of closure assertions, the answer to `C` in `E` should be the set `{d}`, as `d` is the only closed individual in `E`. In order to achieve this goal by means of epistemic queries, `C` should be broken into two assertion queries $\texttt{C}_1$ and $\texttt{C}_2$, given by:

$$\begin{aligned}\texttt{C}_1 &= \neg\texttt{Book(a)}\\ \texttt{C}_2 &= \neg\mathbf{K}\texttt{Book(d)}.\end{aligned}$$

The transformation is cumbersome and, when applied to nested concepts, is prone to generate a number of queries which is exponential in the number of individuals in the document base. In addition, it can be performed only once the closed individuals are known. But then, it is preferable to use closure assertions in a more direct and neat way, devising a semantics that reflects the intuition behind these assertions. And this is precisely our approach.

4 Document bases

We assume two alphabets[2] of symbols, called *primitive concepts* and *primitive roles*. The letter A will denote a primitive concept. Further, we assume an alphabet $\mathcal{O}$ of symbols called *individuals*, denoted by a and b, and an alphabet $\mathcal{V}$ of *variables* (denoted by x and y). The alphabet of *objects*, written $\mathcal{O}^+$, is $\mathcal{O}^+ = \mathcal{O} \cup \mathcal{V}$, (objects are denoted by w and z). The *concepts* (denoted by the letters C and D) and the *roles* (denoted by the letter R) of the language $\mathcal{ALC}^{cl}$ are formed out of primitive concepts and roles according to the following syntax rule:

$$\begin{array}{lrl} C, D \quad \longrightarrow & \top \mid & \text{(top concept)}\\ & \bot \mid & \text{(bottom concept)}\\ & A \mid & \text{(primitive concept)}\\ & C \sqcap D \mid & \text{(concept conjunction)}\\ & C \sqcup D \mid & \text{(concept disjunction)}\\ & \neg C \mid & \text{(concept negation)}\\ & \forall R.C \mid & \text{(universal quantification)}\\ & \exists R.C & \text{(existential quantification)}\end{array}$$

Roles in $\mathcal{ALC}^{cl}$ are always primitive. As customary, in the rest of this document we will omit parentheses around concepts, unless the need for disambiguation arises.

An *assertion* is an expression having one of the following forms:

- $C(w)$, meaning that w is an instance of C, where w is an object and C is a $\mathcal{ALC}^{cl}$ concept;
- $R(w, z)$, meaning that w is related to z by means of R, where w and z are objects and R is a $\mathcal{ALC}^{cl}$ role.

An assertion made out of a primitive symbol is called *primitive assertion*. An assertion made out of a negated primitive symbol is called *negated primitive assertion*. An *ABox* is a finite set of assertions.

Let a be an individual, let R be a primitive role and let T be a primitive concept or primitive role. An *individual closure* is an expression of type $\texttt{Cl}(a)$. The individual a is said to be *closed*. A *primitive closure* is an expression of type $\texttt{Cl}(T)$. The term T is said to be *closed*. A *CBox* is a finite set of closures. An $\mathcal{ALC}^{cl}$ knowledge base is a pair (Σ, Ω), where Σ is an ABox and Ω is a CBox.

Let Δ be the *domain*, a countable infinite set of symbols, called *parameters* and denoted by p_1 and p_2, and γ a fixed injective function from $\mathcal{O}$ to Δ. An *interpretation* $\mathcal{I}$ is a total

[2] In the following, all alphabets are pairwise disjoint.

function mapping: every object into Δ such that for all individuals a, $a^{\mathcal{I}} = \gamma(a)$; every concept to a subset of Δ and every role to a subset of $\Delta \times \Delta$, so that the following equations are satisfied:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta \\
\bot^{\mathcal{I}} &= \emptyset \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta - C^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{p_1 \in \Delta \mid \text{ for all } p_2, \\
&\quad (p_1, p_2) \in R^{\mathcal{I}} \text{ implies } p_2 \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{p_1 \in \Delta \mid \text{ there exists } p_2, \\
&\quad (p_1, p_2) \in R^{\mathcal{I}} \text{ and } p_2 \in C^{\mathcal{I}}\}
\end{aligned}$$

Satisfaction of assertions is defined in the standard way, i.e. an interpretation $\mathcal{I}$ satisfies $C(w)$ if and only if $w^{\mathcal{I}} \in C^{\mathcal{I}}$, whereas $\mathcal{I}$ satisfies $R(w, z)$ if and only if $(w^{\mathcal{I}}, z^{\mathcal{I}}) \in R^{\mathcal{I}}$. Finally, $\mathcal{I}$ satisfies, or is a model of, an ABox if it satisfies each assertion in the set.

Satisfaction of closures is defined on the basis of a notion of minimal knowledge, modelled by *epistemic interpretations.* An epistemic interpretation is a pair $(\mathcal{I}, \mathcal{W})$, where $\mathcal{I}$ is an interpretation and $\mathcal{W}$ is a set of interpretations. An epistemic interpretation *satisfies an individual closure* $\mathtt{Cl}(a)$ if and only if the following two conditions hold:

1. for every primitive concept symbol A, $a^{\mathcal{I}} \in A^{\mathcal{I}}$ if and only if $a^{\mathcal{I}} \in A^{\mathcal{J}}$ for any $\mathcal{J} \in \mathcal{W}$;
2. for every primitive role symbol R and parameter $p \in \Delta$, $(a^{\mathcal{I}}, p) \in R^{\mathcal{I}}$ if and only if $(a^{\mathcal{I}}, p) \in R^{\mathcal{J}}$ for any $\mathcal{J} \in \mathcal{W}$.

As it will be clear in a moment, in an epistemic interpretation $(\mathcal{I}, \mathcal{W})$, the elements of $\mathcal{W}$ are supposed to represent all the possible worlds that the modelled agent (the document base) is aware of, thus forming the context in which closures are to be interpreted. In particular, if a is a closed individual, condition 1 above allows $a^{\mathcal{I}}$ to belong only in the extensions of the primitive concepts that contain $a^{\mathcal{I}}$ in all possible worlds. Condition 2 imposes an analogous constraint on extensions of primitive roles.

An epistemic interpretation *satisfies a primitive closure* $\mathtt{Cl}(A)$ if and only if the following condition hold:

1. for every parameter $p \in \Delta$, $p \in A^{\mathcal{I}}$ if and only if $p \in A^{\mathcal{J}}$ for any $\mathcal{J} \in \mathcal{W}$.

An epistemic interpretation *satisfies a primitive closure* $\mathtt{Cl}(R)$ if and only if the following condition hold:

1. for every parameter $p, p' \in \Delta$, $(p, p') \in R^{\mathcal{I}}$ if and only if $(p, p') \in R^{\mathcal{J}}$ for any $\mathcal{J} \in \mathcal{W}$.

An epistemic interpretation *satisfies* (is a *model* of) a CBox if and only if it satisfies each closure in the set.

An interpretation $\mathcal{I}$ is a *model* of (Σ, Ω) if and only if $\mathcal{I}$ is a model of Σ and $(\mathcal{I}, \mathcal{M}(\Sigma))$ is a model of Ω, where $\mathcal{M}(\Sigma)$ are the models of Σ. It is easy to verify that, for any model of a knowledge base and closed individual a, $a^{\mathcal{I}}$ is allowed in the extension of a primitive concept A just in case $A(a)$ is a logical consequence of the assertions, in symbols $\Sigma \models A(a)$. Analogously, a model $\mathcal{I}$ of (Σ, Ω) is a model of Σ in which, for every primitive role R, closed individual a and individual b, $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ if and only if $\Sigma \models R(a, b)$. In other words, closures force minimal knowledge on closed individuals, ruling out models in which these individuals show up in undue places.

A knowledge base (Σ, Ω) *logically c-implies* a simple assertion α, in symbols $(\Sigma, \Omega) \models_c \alpha$, if and only if every model of (Σ, Ω) satisfies α.

Next section investigates the main properties of the model introduced so far. Proofs, omitted for space reason, can be found in [12].

5 Properties of the Model

First, let us consider the example introduced in Section 2, having:

$$\begin{aligned}
\Sigma &= \{\mathtt{Letter(d)}, \mathtt{Sender(d,Guglielmo)}, \\
&\quad \mathtt{Scottish(Guglielmo)}, \mathtt{Letter(d1)}\} \\
\Omega &= \{\mathtt{Cl(d)}, \mathtt{Cl(Book)}\}
\end{aligned}$$

Thanks to the closure of `d`, in all the models of (Σ, Ω), $\mathtt{d}^{\mathcal{I}}$ only belongs to the extension of `Letter` and, as first member of a pair, to that of `Sender`. The other parameters of Δ, among which there are $\mathtt{d1}^{\mathcal{I}}$ and $\mathtt{Guglielmo}^{\mathcal{I}}$, are free to occur in any of the extensions of the primitive concepts, different from `Book`, and roles, provided, of course, that all the assertions in Σ are satisfied. Similarly, thanks to the closure of `Book`, in all the models $\mathcal{I}$ of (Σ, Ω), $\mathtt{Book}^{\mathcal{I}} = \emptyset$. Among other things, this means that:

- in all the models of (Σ, Ω), $\mathtt{d1}^{\mathcal{I}}$ is not in the extension of `Book`, hence, as desired,
$$(\Sigma, \Omega) \models_c \neg\mathtt{Book(d1)};$$
- in all the models of (Σ, Ω), $\mathtt{Guglielmo}^{\mathcal{I}}$ is the only parameter to be in the extension of `Sender` as a second member of a pair whose first element is $\mathtt{d}^{\mathcal{I}}$; moreover, in all the models of (Σ, Ω), $\mathtt{Guglielmo}^{\mathcal{I}}$ is in the extension of `Scottish`; it follows that, all the models of (Σ, Ω) satisfy $\forall$`Sender.Scottish(d)`, therefore:
$$(\Sigma, \Omega) \models_c \forall\mathtt{Sender.Scottish(d)}.$$

Thus, in order to achieve his goal, Adso would have just to close `d` and `Book`.

More generally, a closure completes the knowledge on the closed individual, granting inferences requiring negative information that normally would not be licensed. As a consequence, for every concept of the language, either the closed individual is an instance of that concept or it is an instance of the negation of that concept.

A concept C is said to be *quantifier free* if no quantifier occurs in it. Moreover, a knowledge base is called *completely closed w.r.t. individuals* if all individuals appearing in it are closed; a knowledge base is called *completely closed w.r.t. primitives* if all primitives appearing in it are closed; a knowledge base is called *completely closed* iff it is completely closed w.r.t. individuals or it is completely closed w.r.t. primitives.

In classical logic, a theory is said to be complete if, for any sentence α, either α or its negation follows from the theory. The next Proposition shows that closing an individual (or a primitive) amounts to make the knowledge about it complete in the classical sense. Since an assertion containing a quantifier involves also other individuals, a proviso is required in the first part of the next theorem. The second part shows that, when all the individuals are closed, the proviso is no longer needed.

Proposition 1 *Let (Σ, Ω) be a knowledge base, $\mathtt{Cl}(a) \in \Omega$, and $C(a)$ a concept assertion. Then:*

1. *either $(\Sigma, \Omega) \models_c C(a)$ or $(\Sigma, \Omega) \models_c \neg C(a)$, for any quantifier free C;*
2. *if (Σ, Ω) is completely closed w.r.t. individuals, then either $(\Sigma, \Omega) \models_c C(a)$ or $(\Sigma, \Omega) \models_c \neg C(a)$, for any C.* ∎

It is natural to ask how c-implication relates to classical logical implication, which is denoted as $\models$. The answer to this question comes in three steps. First, a knowledge base with no closures is equivalent to a set of simple assertions, in that the two have the same models, *i.e.* an interpretation $\mathcal{I}$ is a model of $(\Sigma, \emptyset)$ if and only if $\mathcal{I}$ is a model of Σ.

Second, in presence of closures, c-implication extends classical implication, that is $\models \subset \models_c$. This relationship is derived in two steps: next Proposition asserts that $\models \subseteq \models_c$.

Proposition 2 *Let (Σ, Ω) be a knowledge base and $C(a)$ a simple assertion. Then $\Sigma \models C(a)$ implies $(\Sigma, \Omega) \models_c C(a)$.* ∎

In order to show that $\models \neq \models_c$, it suffices to consider the knowledge base (Σ, Ω) defined at the beginning of this Section. As it has been shown, $\Sigma \not\models \neg\mathtt{Book}(\mathtt{d1})$, whereas $(\Sigma, \Omega) \models_c \neg\mathtt{Book}(\mathtt{d1})$.

Finally, the next Proposition shows exactly what is the inferential gain of c-implication over classical implication.

Proposition 3 *Let (Σ, Ω) be a knowledge base and $\mathtt{Cl}(a) \in \Omega$. Then for each primitive concept A,*

1. *$\Sigma \models A(a)$ implies $(\Sigma, \Omega) \models_c A(a)$;*
2. *$\Sigma \not\models A(a)$ implies $(\Sigma, \Omega) \models_c \neg A(a)$.*

Conversely, if (Σ, Ω) is satisfiable, then for each primitive concept A,

3. *$(\Sigma, \Omega) \models_c A(a)$ implies $\Sigma \models A(a)$;*
4. *$(\Sigma, \Omega) \models_c \neg A(a)$ implies $\Sigma \not\models A(a)$.* ∎

In fact, part 1 of the last Proposition is a special case of Proposition 2 and it has been stated in this form for symmetry with the rest of the Proposition. It states that c-implication preserves implication. Part 2 further confirms the adequacy of our model, by stating that the absence of positive, primitive information on closed individuals is understood as evidence to the contrary. But the most important part of the Proposition is the converse, as it establishes the consequences of c-implication. If (Σ, Ω) is not satisfiable, everything is c-implied by it, so the case is not particularly interesting, in this context. On the contrary, if (Σ, Ω) is satisfiable, the positive primitive assertions c-implied by it are exactly those implied by Σ, whereas the negative primitive assertions c-implied by it are exactly the negation of the positive, primitive assertions not implied by Σ.

The last result gives us the possibility of comparing our model with Naive CWA, historically the first notion of CWA to be proposed. Naive CWA is defined for finite sets of first-order sentences without equality and whose prenex normal forms contain no existential quantifiers. If T is one such sets, then the naive closure of T, *NCWA(T)*, is given by:

$$NCWA(T) = T \cup \{\neg A : T \nvdash A \text{ and } A \in HB(T)\},$$

where $HB(T)$ is the Herbrand Base of T. Given the completeness of first-order logic, we may replace $\vdash$ by $\models$. Furthermore, since we are considering theories with no function symbols, the correspondent of $HB(T)$ is the set of positive primitive assertions. It then follows that c-implication captures Naive CWA for theories which, although not as expressive as a first-order calculus, do provide the basic connectives and both quantifiers, in some form.

6 A calculus for instance checking

In this section we will briefly describe a constraint propagation based calculus solving $(\Sigma, \Omega) \models_c C(a)$. For simplicity we introduce a new kind of concept operator, called *one-of enumeration*, of the following form $\{a_1, \ldots, a_n\}$, where the a_i are individuals and $n \geq 0$. From a semantics point of view, given an interpretation $\mathcal{I}$, the extension of $\{a_1, \ldots, a_n\}$, denoted by $\{a_1, \ldots, a_n\}^{\mathcal{I}}$, is simply the set $\{a_1{}^{\mathcal{I}}, \ldots, a_n{}^{\mathcal{I}}\}$.

The calculus consists of similar rules as those used to test the satisfiability of an $\mathcal{ALC}$ KB and is in fact an extension of the one presented in [16].

It is worth noting that the method usees the following properties: $\Sigma \models C(a)$ if and only if the ABox $\Sigma \cup \{\neg C(a)\}$ has no model, if and only if no clash free constraint system (a model) can be build by the application of the rules, from the ABox $\Sigma \cup \{\neg C(a)\}$.

Essentially, in our calculus a clash arises either when a classical clash does ($\perp (w)$ is obtained or both $A(w)$ and $\neg A(w)$, for some A and w), or when some positive, primitive constraint shows up to violate the constraints on epistemic interpretations defined previously, *e.g.* a violation takes place in a constraint system S as soon as a constraint appears which

binds a closed individual a to a primitive concept symbol A such that $A(a)$ is not logically implied by the KB. In this case, the insertion of $A(a)$ in S would result in the construction of a model in which a belongs to the extension of A while there are models in which this does not happen.

A *constraint* is an assertion. Let (Σ, Ω) be a KB. A *constraint system* is a set S (ABox) of constraints such that $\Sigma \subseteq S$.

Let S be a constraint system, and let (Σ, Ω) be a KB. An interpretation $\mathcal{I}$ (Σ, Ω)-*satisfies* S if $\mathcal{I}$ satisfies all of its constraints and $(\mathcal{I}, \mathcal{M}(\Sigma))$ satisfies Ω: $\mathcal{I}$ is called a (Σ, Ω)-*model* of S. S is said (Σ, Ω)-*solvable* if there is a (Σ, Ω)-*model* of S, otherwise S is said (Σ, Ω)-*unsolvable*.

As first, we solve the (Σ, Ω)-solvability problem of a constraint system S. Clashes are defined as follows:

Definition 1 *A constraint system S contains a (Σ, Ω)-clash iff at least one of the following conditions hold:*

1. *S contains a constraint of the form: $\bot(w)$;*
2. *S contains two constraints of the form: $A(w), \neg A(w)$;*
3. *S contains a constraint of the form: $\{a_1, \ldots, a_n\}(a)$ and $a \neq a_i$ for all $i = 1, \ldots, n$;*
4. *S contains a constraint of the form: $\neg\{a_1, \ldots, a_n\}(a)$ and $a = a_i$ for some $i = 1, \ldots, n$;*
5. *S contains a constraint of the form: $\{\}(w)$;*
6. *S contains a constraint of the form $A(a)$ such that $\mathtt{Cl}(a) \in \Omega$ or $\mathtt{Cl}(A) \in \Omega$, and $\Sigma \not\models A(a)$;*
7. *S contains a constraint of the form $\neg A(a)$ such that $\mathtt{Cl}(a) \in \Omega$ or $\mathtt{Cl}(A) \in \Omega$, and $\Sigma \models A(a)$;*
8. *S contains a constraint of the form $R(a, b)$ such that $\mathtt{Cl}(a) \in \Omega$ or $\mathtt{Cl}(R) \in \Omega$, and $\Sigma \not\models R(a, b)$.* ∎

Please note that in $\mathcal{ALC}$, only clashes of type *1.* and *2.* arise.

In order to check the (Σ, Ω)-solvability of a constraint system S, the following set of *completion rules* are used: let (Σ, Ω) be a knowledge base, let $\mathcal{O}^S$ be the set of individuals occurring in a constraint set S, let A be a primitive concept, let R be a role, let O be an enumerated one-of set, let $S[x/a]$ be the constraint set S in which all occurrences of variable x are substituted with the individual a and let

$$\begin{array}{rcl} O^i & = & \mathcal{O}^S \cup \{i\} \text{ where } i \text{ is a new individual} \\ O_A & = & \{a \in O^i : \Sigma \models A(a)\} \\ O_{\neg A} & = & O^i \setminus O_A \\ O_R & = & \{(a, b) \in O^i \times O^i : \Sigma \models R(a, b)\} \\ O_{\neg R} & = & O^i \times O^i \setminus O_R \\ O^1 & = & \{a : (a, b) \in O\} \\ O^2 & = & \{b : (a, b) \in O\} \end{array}$$

The rules are the following:

1. $S \rightarrow_{\sqcap}$ $S \cup \{C(w), D(w)\}$
   if $(C \sqcap D)(w)$ is in S, and $C(w)$ and $D(w)$ are not both in S;
2. $S \rightarrow_{\sqcup}$ $S \cup \{E(w)\}$
   if $(C \sqcup D)(w)$ is in S, neither $C(w)$ nor $D(w)$ is in S and $E = C$ or $E = D$;
3. $S \rightarrow_{\exists}$ $S \cup \{R(w, x), C(x)\}$
   if $(\exists R.C)(w)$ is in S, there is no z such that both $R(w, z)$ and $C(z)$ are in S;
4. $S \rightarrow_{\forall}$ $S \cup \{C(z)\}$
   if $(\forall R.C)(w)$ is in S, $R(w, z)$ is in S and $C(z)$ is not in S;
5. $S \rightarrow_{[/]}$ $S[x/a_i]$
   if $\{a_1, \ldots, a_n\}(x)$ is in S, and $i \in \{1, \ldots, n\}$
6. $S \rightarrow_{\mathtt{Cl}^1_R}$ $S \cup \{O^2_R(x)\}$
   if $\mathtt{Cl}(w) \in \Omega$, or $\mathtt{Cl}(R) \in \Omega$, and $R(w, x)$ is in S and $O^2_R(x)$ is not in S;
7. $S \rightarrow_{\mathtt{Cl}^2_R}$ $S \cup \{O^1_R(x)\}$
   if $\mathtt{Cl}(R) \in \Omega$ and $R(x, w)$ are in S and $O^1_R(x)$ is not in S;
8. $S \rightarrow_{\mathtt{Cl}_A}$ $S \cup \{O_A(x)\}$
   if $\mathtt{Cl}(A) \in \Omega$, $A(x)$ is in S and $O_A(x)$ is not in S;
9. $S \rightarrow_{\mathtt{Cl}_{\neg A}}$ $S \cup \{O_{\neg A}(x)\}$
   if $\mathtt{Cl}(A) \in \Omega$, $\neg A(x)$ is in S and $O_{\neg A}(x)$ is not in S. ∎

Again, please note that for $\mathcal{ALC}$ only rules *1.* – *4.* are used.

A constraint system is said to be *complete* if no rule is applicable to it. Any complete constraint system obtained from a constraint system S by applying the above rules is called a *completion* of S. Note that, due to the presence of the rules $\rightarrow_{\sqcup}$, $\rightarrow_{[/]}$, more than one completion can be obtained starting from a constraint system. Rules $\rightarrow_{\sqcup}$, $\rightarrow_{[/]}$ are called *nondeterministic* rules. All other rules are called deterministic rules.

Proposition 4 *Let (Σ, Ω) be a KB, and S, S' be two constraint systems. Then:*

1. *If S' is obtained from S by application of one of deterministic rule, then S is (Σ, Ω)-solvable iff S' is (Σ, Ω)-solvable;*
2. *If S' is obtained from S by application of one of a nondeterministic rule, then S is (Σ, Ω)-solvable if S' is (Σ, Ω)-solvable. Conversely, if S is (Σ, Ω)-solvable and one of the nondeterministic rules is applicable, then one of it can be applied in such a way that it yields a (Σ, Ω)-solvable constraint system.* ∎

Proposition 5 *Let S be a constraint system and (Σ, Ω) be a KB. Then S is (Σ, Ω)-solvable iff there exists at least one completion of S that contains no clash.* ∎

Proposition 6 *Let (Σ, Ω) be a KB. Then $(\Sigma, \Omega) \models_c C(a)$ iff $\Sigma \cup \{a : \neg C\}$ is (Σ, Ω)-unsolvable.* ∎

From a computational complexity point of view, since checking whether $\Sigma \models C(a)$ is PSPACE-complete for $\mathcal{ALC}$[16] and since $\Sigma \models C(a)$ if and only if $(\Sigma, \emptyset) \models_c C(a)$, it follows that checking $(\Sigma, \Omega) \models_c C(a)$ is a PSPACE-hard problem. Nevertheless, checking $(\Sigma, \Omega) \models_c C(a)$ can be done in polinomial

space with a similar table technic as described in [8], Section 4, for answering $\mathcal{ALCK}$ queries. For space reasons we omit it.

Proposition 7 *Let (Σ,Ω) be a KB. Then checking whether $(\Sigma,\Omega) \models_c C(a)$ is PSPACE-complete.* ∎

7 Summary and future work

We have defined a description logic whose assertional language includes closures, expressing meta-knowledge on the knowledge on the involved individuals and primitive concepts. Clearly, modelling multimedia documents requires much more. However, we do think that providing these closures is a necessary step towards the definition of an adequate logic for multimedia retrieval.

In fact, we are currently working on a number of extensions as

- a *Relevance logic based DL* [13] with the aim to (*i*) capture a tight connection in meaning between query and the information that are retrieved in response to it; (*ii*) ruling out undesiderable inferences as paradoxes of logical implication and reasoning by cases; (*iii*) dealing with multimedia document's content descriptions which can be contradictory in nature and, hence, consistency of KB is unreasonable with respect to the data represented, the huge amount of data and unpraticability of belief revision;
- a probabilistic proof procedure for a probabilistic version of DLs, which is essentially in case of dealing with various source of uncertainty, such as uncertainty related to domain knowledge and uncertainty related to automatic document representation.

REFERENCES

[1] Alessandro Artale and Enrico Franconi, 'Hierarchical plans in a description logic of time and action', in *International Workshop on Description Logics*, eds., Borgida A., Lenzerini M., Nardi D., and Nebel B., pp. 1–5, Rome, Italy, (1995).

[2] Franz Baader and Philipp Hanschke, 'A schema for integrating concrete domains into concept languages', in *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pp. 452–457, Sydney, (1991).

[3] Franz Baader and Armin Laux, 'Terminological logics with modal operators', in *International Workshop on Description Logics*, eds., Borgida A., Lenzerini M., Nardi D., and Nebel B., pp. 6–12, Rome, Italy, (1995).

[4] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori, 'Using subsumption in semantic query optimization', in *IJCAI Workshop on Object-Based Representation Systems*, ed., A. Napoli, (1993).

[5] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf, 'Decidable reasoning in terminological knowledge representation systems', in *Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI-93)*, pp. 704–709, Chambery, France, (1993). Morgan Kaufmann, Los Altos.

[6] F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf, and W. Nutt, 'Adding epistemic operators to concept languages', in *Proceedings of KR-92, the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pp. 342–353, (1992).

[7] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt, 'Tractable concept languages', in *Proceedings of IJCAI-91, 12th International Joint Conference on Artificial Intelligence*, pp. 458–463, Sidney, Australia, (1991).

[8] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf, 'Adding epistemic operators to concept languages', in *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-92)*, pp. 342–353. Morgan Kaufmann, Los Altos, (1992).

[9] Alon Y. Levy and Marie-Christine Rousset, 'CARIN: A representation language combining horn rules and description logics', in *International Workshop on Description Logics*, eds., Borgida A., Lenzerini M., Nardi D., and Nebel B., pp. 44–51, Rome, Italy, (1995).

[10] W. Lukaszewicz, *Non-Monotonic reasoning*, chapter 7: Approaches to Closed World Assumption, 281–308, Ellis Horwood series in Artificial Intelligence, Ellis Horwood, New York, 1990.

[11] C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos, 'A model of information retrieval based on a terminological logic', in *Proceedings of SIGIR-93, 16th ACM Conference on Research and Development in Information Retrieval*, pp. 298–307, Pittsburgh, PA, (July 1993).

[12] C. Meghini and U. Straccia, 'A description logic for multimedia information retrieval', Technical report, Consiglio Nazionale delle Ricerche, Istituto di Elaborazione della Informazione, (forthcoming).

[13] Peter F. Patel-Schneider, 'A four-valued semantics for terminological logics', *Artificial Intelligence*, **38**, 319–351, (1989).

[14] R. Reiter, 'On asking what a database knows', in *Proceedings of the Symposium on Computational Logic*, ed., J.W. Lloyd, Commission of the European Communities, DG XIII, Esprit Basic Research Series, 96–113, Springer Verlag, (Nov 1990).

[15] Raymond Reiter, 'On closed world data bases', in *Logic and data bases*, eds., Hervé Gallaire and Jack Minker, 55–76, Plenum Press, New York, NY, (1978).

[16] Manfred Schmidt-Schauß and Gert Smolka, 'Attributive concept descriptions with complements', *Artificial Intelligence*, **48**, 1–26, (1991).

[17] Albrecht Schmiedel, 'A temporal terminological logic', in *Proceedings of AAAI-90, 8th Conference of the American Association for Artificial Intelligence*, pp. 640–645, Boston, MA, (1990).

[18] Fabrizio Sebastiani, 'A probabilistic terminological logic for modelling information retrieval', in *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pp. 122–130, Dublin, IRL, (1994). Published by Springer Verlag, Heidelberg, FRG.

[19] Umberto Straccia, 'Default inheritance reasoning in hybrid KL-ONE-style logics', in *Proceedings of IJCAI-93, 13th International Joint Conference on Artificial Intelligence*, pp. 676–681, Chambery, France, (1993).

[20] John Yen, 'Generalizing term subsumption languages to fuzzy logic', in *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pp. 472–477, Sydney, Australia, (1991).