Effective and Efficient Relational Query Processing Using Conceptual Graphs

Iadh Ounis and Marius Paşca

CLIPS-IMAG, MRIM Team, University of Grenoble

BP 53, 38041 Grenoble Cedex, France

Abstract

In a recent paper (see Ounis & Huibers's work), a logical relational framework for information retrieval is presented, which emphasizes the importance of relations for accurate indexing, as well as the need to provide relation treatments for effective retrieval. Most knowledge representation formalisms support relational indexing. However, the majority of them do not fully allow relation treatment. Conceptual graphs, though offering the richness needed to use relations in the indexing of complex and highly structured documents, do not provide further relation-based processing. In this paper, we follow the above relational framework in the case of conceptual graphs, such that we can take into account relation properties in the retrieval process. The approach leads to a sound extension which preserves the semantics of this formalism. In its implementation, an important goal was to obtain a workable system. Experimental results prove not only improvement in retrieval effectiveness, but also good execution time.

1 Introduction

It is often the case, in the indexing processes used by information retrieval systems, that an important amount of the data remains unused (or unread) simply because there are no means for retrieving this information effectively. This is usually due to the limits of the formalism chosen to describe the document contents. The most intricate or carefully designed retrieval algorithm cannot compensate for inappropriate representations of documents. The accuracy of representation for a document (a text, an image, sounds and the like) is determined by the formalism used for indexing. The power of the formalism gives the difference between the information content of a document and the part that is extracted and represented according to the formalism.

On the other side, an information retrieval system is evaluated according to several criteria, and speed is one of the important ones [1]. If a retrieval system is too slow, it will be unusable, regardless of its ability to identify relevant documents. One of the characteristics of typical users is that they do not like to wait for the answers, and recent trends in the volume and availability of information suggest that system speed will only become more important.

Classical information retrieval systems use keywords, a choice that allows for fast retrieval. The disadvantages of this approach consist in severe limitations of the way in which the user can find or use the information he needs, as keywords are not sufficient for effectively directing a search process. As a simplistic example, the result of a search on the Net (using Altavista, Yahoo or other search engines) is quickly obtained but sometimes consists of such a large

collection of documents that the speed serves to nothing. Indeed, a list of about 1000 documents retrieved for a single query would probably not be exploited efficiently by the typical user. The need of an expressive indexing language for new applications handling complex and multimedia documents has been mentioned in [2, 3, 4]. In [5], it is argued that only languages supporting relational indexing can be used in multimedia information retrieval.

As indexing language, we choose here the conceptual graphs formalism, which is expressive enough to represent accurate and highly structured information and fulfills the need of relational indexing. Conceptual graphs have particularly gained attention in information retrieval (IR) because they seem to better model the domains of some new applications of IR such as digital libraries and hypermedia systems [6, 7, 8]. Like all other expressive languages, its operators are complicated and time-demanding. In particular, previous implementations of the operator used to implement the matching function in *CGs* have shown its expensiveness [9, 10]. Another problem of conceptual graphs is that they do not provide relation-based reasoning. Indeed, recent studies about the impact of structured and multimedia documents on both indexing and retrieving showed the need to represent and to exploit relationships between terms [3, 5, 11, 12]. In previous research, following a relational indexing approach, Ounis & Huibers suggest a prime use of the relation properties through a logical framework, in such a way that it can improve the effectiveness of the matching function [13]. This logical framework can be applied to all the knowledge representation formalisms that don't allow for relation-based reasoning, and we instantiate it to the case of *CGs*.

Therefore, the aim of this paper is twofold. Firstly, it provides an instance to the relational indexing framework in [13] for the case of conceptual graphs. We propose a relational extension that makes it possible to reason on relations on a sound basis, thus improving retrieval effectiveness. Secondly, we propose an efficient matching function for conceptual graphs that preserves the semantics of the formalism as introduced by Sowa [14], while additionally taking into account relational reasoning as part of indexing. Retrieval time is thus reduced, even though it incorporates relation-based reasoning.

The paper is organized as follows. In section 2, we introduce the need for relational indexing and reasoning in information retrieval. In section 3, we present the conceptual graph formalism. Section 4 introduces a sound extension for this formalism, allowing for relation-based reasoning. We present a way to introduce reasoning through derivations on conceptual graphs, allowing for relation treatment in section 5. In section 6, we give a way to implement the approach in order to achieve efficient retrieval. Section 7 presents the experimental results of our implementation. Section 8 gives a conclusion on the impact of the work presented in this paper.

2 Relations in Information Retrieval

As mentioned in the previous section, keywords are a very popular formalism in the information retrieval. They have a few net advantages, such as simplicity and practicability. Fast algorithms are available for keyword-based retrieval, hence their application in commercial search engines [15]. Unfortunately, keywords are not sufficient to accurately represent non-textual documents, such as images.

Indeed, as an illustration, assume that we would like to index the image of figure 1. If only keywords are used, the information content of this image could be indexed by the set {Man, Chair, Table}. However, for an homogeneous image collection, composed for instance of portraits such as the one presented here, it would be improbable that all the users be interested only in images showing certain objects, e.g. a man and a table. Here are two examples of what

other things they would be likely to search for. Firstly, the relative geographic position of objects in the image is also important, to support queries such as *a man on the left of a table*. We model it in the image index by the spatial relation *on the left* between the objects *man* and *table*. Secondly, semantic refinements would be very helpful, to answer to queries such as *a man seated on a chair*. One can model this by including a relation *seated-on* between the *man* and the *chair* in the index. We note that modeling relations as keywords is not an appropriate solution, because retrieval effectiveness would be degraded, due to an increased noise. For instance, if the relation *on the left* were represented as a keyword added to the set {Man, Chair, Table}, then one could not say between which keywords the relation holds, and the image would be incorrectly indexed.

Figure 1: The photograph of Gabriel Faure is an illustration of the importance of relations in index quality

Moreover, let us consider now that our index is composed of keywords and relations, {Man, Chair, Table, Man *on the left of* Table, Man *seated on* Chair}. A characteristic of the indexing process is that it does not generally include information that is considered to be redundant, for obvious storage efficiency reasons. However, adding such information could have an important impact on retrieval effectiveness. Hence, relation-based reasoning is very useful. In particular, if we represent relation properties and then consider them in the matching function, effectiveness can be much improved. As an illustration, every time that an object is on the left of another, for instance the man who is on the left of the table, the inverse relation also holds in the opposite direction, i.e. the table is on the right of the man. As one cannot predict which spatial relation the user will be interested in and introduce in the query, these treatments are required to retrieve additional relevant images. These types of treatments should be accurately modeled by the formalism used for indexing, as mentioned in [13].

3 Conceptual Graphs: an Expressive Formalism for Information Retrieval

In order to have the support to represent complex, relation-based information, we use conceptual graphs as the indexing language. Conceptual graphs (*CGs*) are an expressive knowledge representation formalism introduced by Sowa [14].

Figure 2: A simple conceptual graph, for “The man named Gabriel Faure is seated on a chair”

A conceptual graph is an oriented graph that consists of concept nodes, conceptual relation nodes or simply relation nodes, and edges between concept and relation nodes [14]. Concept nodes represent entities, attributes, states and events, and relation nodes show how the concepts are interconnected. A concept, represented graphically by a box, has a *type* (which corresponds to a semantic class) and possibly a *referent* (which corresponds to an instantiation to an individual of the class). For instance, $\boxed{\text{MAN} : *}$ stands for the concept of all possible men. This concept is called a *generic* concept also noted $\boxed{\text{MAN}}$. On the other hand, $\boxed{\text{MAN} : \text{Gabriel Faure}}$ obviously stands for the concept of a man named Gabriel Faure. A relation, represented graphically by an oval, has only a type. The number of concepts linked by a relation must be equal to the arity of its relation type.

The box and oval notations are called the *display form* [14]. For the sake of simplicity, the *linear form* may be also employed in the *CG* notation. For instance, the same conceptual graph can be represented either in the display form shown in figure 2 or in its equivalent linear form: $[\text{MAN} : \text{Gabriel Faure}] \rightarrow (\text{Seated-on}) \rightarrow [\text{CHAIR}]$.

Figure 3: There is a projection of the query $\chi(q)$ in the simplified index of the image in figure 1, $\chi(d)$. The result of the projection is shown by a dotted line. Conversely, we have a logical implication from the document to the query.

Let us use the notation d for the image document in figure 1. We could model it by a conceptual graph, whose simplified index could be $\chi(d)$ in figure 3. As one can see, objects, as well as relations holding between them, can be easily modeled by the formalism. The arrow directions provide a way to read the information in the conceptual graph. In this case, we have a table and a man identified as Gabriel Faure, who is seated on a chair. In addition, the back of the chair, which is a part of the chair, is on the left of the man and the man is on the left of the table.

The formalism is provided with four graph construction operators [14]. For instance, the so-called *join* operator allows to join together two graphs, each corresponding to some piece of information. A bigger graph is obtained from the two smaller ones, by connecting the latter on their common concepts. In our sample image, the man who is seated on the chair is the same man who is on the left of the table. Therefore, the graph $\chi(d)$ could be obtained by a join of the two smaller corresponding graphs, each modeling one piece of information about our man, on the concept $[\text{MAN} : \text{Gabriel Faure}]$.

Concept types are organized into a lattice, based on the relation “A-Kind-Of”. This relation is a partial-ordering

relation, which can be interpreted as a categorical *generalization relation*. We note it by $\leq$. For instance, we can model the knowledge that an HUMAN is a generalization of a MAN by the relation $\text{MAN} \leq \text{HUMAN}$. We say that the concept type MAN is a *subtype* of the concept type HUMAN. An example of a concept type lattice is given in figure 4. We note the use of two particular concept types, $\top$ and $\perp$, which are respectively the most general and the most specific concept types in the domain, and always form the extremities of the lattice. Relation types are also formally organized into a lattice in Sowa's formalism, but this lattice is not taken into account by the operators provided with the formalism, as opposed to the case of the concept type lattice [5].

Figure 4: An example of the organization of different concept types into a concept type lattice

Sowa's formalism offers a complete framework for modeling the main components of an IR model, i.e. the document indexes, the query and the matching function [5, 7, 16]. Given a document collection, we have a conceptual graph that indexes each document. For a document d_i , $i \in [1, \text{document collection size}]$, we denote its index in the form of the conceptual graph by $\chi(d_i)$. The query q is also modeled by a conceptual graph $\chi(q)$, and we choose as the matching function a particular operator provided by the formalism, which is called the *projection* operator. The operator permits to compare two conceptual graphs. Informally, in the case of information retrieval it involves searching for a copy of the query graph $\chi(q)$ in the document graph $\chi(d_i)$, up to some concept restrictions. These restrictions are made either on the concept type, by replacing it with one of its subtypes (for instance, replacing [PIECE OF FURNITURE] by [CHAIR]), or on the referent, by instantiating the generic referent $\star$ to a certain individual (for instance, replacing [MAN] by [MAN: Gabriel Faure]), or on both (see figure 3).

An illustration of the use of projection as the matching function is given in figure 3. In this case, we have a projection of the query $\chi(q)$ in the index $\chi(d)$. The projection is a sub-graph of $\chi(d)$, and it is obtained by restricting respectively the concept [MAN] in [MAN: Gabriel Faure] and the concept [PIECE OF FURNITURE] in [CHAIR].

We use conceptual graphs in the context of van Rijsbergen's logical IR model introduced in [17]. In that model, the decision on the relevance of a document for a query is based on the truth of the logical implication of the query from the document. This means that we retrieve a document for a query if we have a logical implication from the document to the query.

The implication is obtained on the basis of the translation to first-order logic, by the ϕ operator given by Sowa in his formalism [14]. For each graph g , it associates a logical formula $\phi(g)$. Now, there is a relation between the existence of a projection between two graphs and their associated formulas. Indeed, it was proven that there exists a

projection of a graph g in a graph h if and only if the formula $\phi(h)$ associated to h implies the formula $\phi(g)$ associated to g [14, 18], $\phi(h) \supset \phi(g)$.

The existence of a projection of $\chi(q)$ in $\chi(d_i)$ is the retrieval decision for d_i in the case of conceptual graphs [16]. In the latter paper, a document d_i of the document collection is retrieved for the query q , if and only if there is a projection of $\chi(q)$ in $\chi(d_i)$. For instance, as a projection exists in figure 3 from $\chi(q)$ in $\chi(d)$, we also have an implication in terms of logical formulas, in the opposite direction, $\phi(\chi(d)) \supset \phi(\chi(q))$.

As one can see, *CGs* offer an intuitive way to model complex documents, including images. In this formalism, the document content can be accurately modeled, not only in terms of objects, but also by the relations that hold between them. The query can be expressed as a conceptual graph as well, thus providing uniformity and allowing for using the projection operator as the matching function. However, in a relational indexing framework, relation-based reasoning is a necessity that is not fulfilled by conceptual graphs [5]. In the following, we present relational extensions to the formalism, which improve the effectiveness of a *CG*-based IR system.

4 Relational Extensions to Conceptual Graphs

In the *CGs* formalism, no operator and in particular the projection operator, bases its processing on knowledge captured by relations. This does not satisfy the general relational indexing framework presented in [13], where we also find arguments for the need of introducing yet another layer, that is relation-based reasoning.

We propose a few extensions to conceptual graphs and instantiate the general framework for the extended formalism. According to the framework, the treatment of relations concerns three kinds of relation properties, that must be captured in the knowledge base of any information retrieval system. Therefore, we take into account the mathematical properties of relation types, the properties about their links to other relations and their behaviour with respect to other relations, as introduced in the mentioned framework. This knowledge is specified as a set of conceptual graphs, obtained by applying restrictions on the concepts of one of the following properties on relation types, represented as well in the form of conceptual graphs:

-

$$[\text{RELATION}] \rightarrow (\text{Attribute}) \rightarrow [\text{PROPERTY}]$$

for the mathematical properties;

-

$$[\text{RELATION}] \rightarrow (\text{Link}) \leftarrow [\text{RELATION}]$$

$$\downarrow$$

$$[\text{RELATION}]$$

for the behaviour of the relations;

-

$$[\text{RELATION}] \rightarrow (\text{SemanticalLink}) \rightarrow [\text{RELATION}]$$

for the semantical links between relations.

We note that *Attribute*, *Link* and *SemanticalLink* constitute second-order relations, as they link not concepts, but relations. They are organized into a separate, second-order type lattice, and may present as well properties (e.g. symmetry) [5].

Relation properties can improve dramatically the effectiveness of retrieval. Their usefulness can be seen for our reference image in figure 1, and for its associated index $\chi(d)$ in figure 3. Let us consider the case of spatial relations for the image. Representing the Left-to relations between the objects allows one to infer, by transitivity, that the back of the chair is on the left of the table. The mathematical property that the relation Left-to is transitive can be represented by the following conceptual graph, obtained by restricting the concepts of the graph given for generic mathematical properties (we suppose that $\text{TRANSITIVITY} \leq \text{PROPERTY}$ in the second-order type lattice):

$$[\text{RELATION} : \text{Left-to}] \rightarrow (\text{Attribute}) \rightarrow [\text{TRANSITIVITY}]$$

This additional knowledge allows to retrieve the image, for a query looking for *a chair and a table, where the back of the chair is on the left of the table*. We assume that the query is put in the form of a conceptual graph: $[\text{TABLE}] \leftarrow (\text{Left-to}) \leftarrow [\text{BACK OF CHAIR}] \rightarrow (\text{Part-of}) \rightarrow [\text{CHAIR}]$. Without relation-based reasoning, there is no projection for this query into $\chi(d)$ of figure 3, because the relation Left-to between the given concepts is missing in $\chi(d)$. Consequently, the image is not retrieved, though it does satisfy the user's query. On the contrary, using the mathematical property of transitivity, which is now represented as part of the domain knowledge, we can derive and “fill in” the missing relation during retrieval, find a projection into the extended index, and retrieve the image for this query. One can see the important impact of this kind of relation properties for spatial relations in general, when the user specifies in its query a particular geographic distribution of the objects in his images of interest.

In the same way, the relations Left-to and Part-of can be linked together in a composition relation, corresponding to the fact that if an object is on the left of another, then any part of the first object is also on the left of the second (we note that $\text{Compose} \leq \text{Link}$ in the second-order type lattice):

$$\begin{array}{c} [\text{RELATION} : \text{Left-to}] \rightarrow (\text{Compose}) \leftarrow [\text{RELATION} : \text{Part-of}] \\ \downarrow \\ [\text{RELATION} : \text{Left-to}] \end{array}$$

Finally, the relations Left-to and the Right-to can be linked by the *Inversion* property. This will correctly retrieve the image for a query looking for *a table on the right of a man*, though the corresponding relation does not appear in the index $\chi(d)$. The relations Left-to and Right-to are equivalent when their arguments are swapped by the *Inversion* property. We note that $\text{Inversion} \leq \text{SemanticalLink}$ in the second-order type lattice:

$$[\text{RELATION} : \text{Left-to}] \rightarrow (\text{Inversion}) \rightarrow [\text{RELATION} : \text{Right-to}]$$

Following [13], each relation type r has a unique signature $\Sigma_r = (r, n, C_1, \dots, C_n)$ which specifies its semantics [5]. Here n is the arity of the relation type r , and $C_1, \dots, C_n$ are the greatest concept types in the concept type lattice which can be linked by the relation type r . Thus the signature imposes a boundary upon the argument concept types. Only specialisations of these concept types can appear as arguments of relations in documents and queries. For example, if we impose that a relation type *Loves* has the signature $\Sigma_{\text{Loves}} = (\text{Loves}, 2, \text{MAN}, \text{WOMAN})$, it would not be possible to have the graph $[\text{MAN}] \rightarrow (\text{Loves}) \rightarrow [\text{HUMAN}]$ in any of the documents, nor in the queries because WOMAN

$\leq$ HUMAN. Formally, we represent a relation signature, in the conceptual graphs formalism, by a *signature graph*. The signature of the above relation type, *Loves*, is represented within *CGs* by the following signature graph:

$$[\text{MAN} : \star] \rightarrow (\text{Loves}) \rightarrow [\text{WOMAN} : \star]$$

Relation signatures are used to model certain constraints between the concepts of the relations involved in a given relation property. For instance, the inversion property applies if the arguments of the relations can be obtained from each other by swapping. Indeed, for the inversion property between relations Left-to and Right-to, if $\Sigma_{\text{Left-to}} = (\text{Left-to}, 2, C_1, C_2)$ and $\Sigma_{\text{Right-to}} = (\text{Right-to}, 2, C'_1, C'_2)$, then $C_1 = C'_2$ and $C_2 = C'_1$.

Moreover, Sowa's basic conceptual graphs model does not take into account the relation of specialisation /generalisation between relation types, nor does it consider order on n-ary relations. In [13] these kinds of properties can be useful in the matching process. Following this idea, in [5] it is shown how relation signatures provide a way to solve this problem, and in particular how they allow for using relations with different arities.

We note that the knowledge contained in the set of properties on relations is associated to a set of derivation rules on the conceptual graphs, one for each relation property. It is such a derivation rule that permits the above derivations of the Left-to relation obtained by transitivity, or of the Right-to relation obtained by inversion.

5 Derivations on Conceptual Graphs

In [13], relation properties are modeled by derivation rules. For instance, the inversion property between two binary relations R_1 and R_2 is modeled by a rule of the form: if $R_1(a, b)$ then $R_2(b, a)$, where a and b are either primitive or more complex indexing terms. In the previous section, we showed that relation properties are modeled in the form of conceptual graphs. For uniformity and soundness of our extension to the formalism, derivation rules are given as pairs of conceptual graphs. For instance, the derivation rule corresponding to the inversion property is given as shown in figure 5. More formally, a derivation rule on graphs has the following definition:

Definition 1 (Derivation rule)

A derivation rule on graphs $label_R: G_1 \Rightarrow G_2$ is a pair of lambda-expressions $(\lambda x_{11} \dots x_{1n} G_1, \lambda x_{21} \dots x_{2n} G_2)$. The notations $x_i, i = 1, n$ constitute variables corresponding to coreference links between concepts occurring simultaneously in G_1 and in G_2 . Hence x_{1i} is a coreference of x_{2i} . The rule is identified by $label_R$.

Figure 5: Derivation rule for the inversion property between Left-to and Right-to

These derivation rules are then applied to the indexes in the document collection, by using the following procedure:

Definition 2 (Applying a Derivation Rule)

A rule $label_R: G_1 \Rightarrow G_2$ applies to a conceptual graph g , if there exists a projection π of G_1 in g . The graph resulted by applying this rule, denoted as $g_{extended}$ (for extended graph), is built from g and G_2 , by joining each variable x_{2i} of G_2 with $\pi(x_{1i})$, where $\pi(x_{1i})$ is the image of the coreference x_{1i} by π . In terms of conceptual graphs operators, the addition consists of the following steps:

- In G_2 restrict all coreference labels x_{2i} to those of $\pi(x_{1i})$
- Join each coreference x_{2i} of G_2 with $\pi(x_{1i})$ in g .

In [5, 19], it is proven that the extension of conceptual graphs by derivation rules is sound and complete, according to Sowa's first-order logic semantics. More details on the integration of derivation rules in IR systems based on *CGs* can be found in [5].

As an illustration of the application of derivation rules, let us consider the derivation rule of figure 5. For the conceptual graph on the left in figure 6, the application of the rule gives the graph on the right, which is obtained by adding the relation Right-to. As derivation rules are applied to all the indexes in the document collection, we end up with a new set of extended indexes, such as the graph on the right in figure 6.

Figure 6: The conceptual graph on the left is extended by adding the Right-to relation. This follows the application of the derivation rule corresponding to the inversion property between Left-to and Right-to.

The impact of this extension on retrieval effectiveness can be illustrated by considering the query $[TABLE] \rightarrow (Right-to) \rightarrow [MAN: Gabriel Faure]$. This query cannot be projected in the non-extended index, which is the left graph in figure 6. By extending the index with the derivation rule for the inversion property, the query can now be projected. In terms of retrieval evaluation, this means that recall and precision are improved.

We may conclude that relational extensions for conceptual graphs improve retrieval effectiveness. Indeed, we presented practical examples of the impact of the relation-based reasoning on the well-known precision /recall evaluation method, while preserving the accuracy obtained without relation-based reasoning. Nevertheless, this introduces a different kind of problem, related to retrieval efficiency. The extension of the projection operator is possible but not applicable in information retrieval. Indeed, retrieval time performance of systems based on *CGs* is poor, due to the computational cost of the projection operator [5, 9, 20, 21]. If other computations are added on top of the already expensive ones, we would get to no practicable solution. Instead of this approach, we perform the derivations as part of indexing, with practically no impact on retrieval performance.

6 Efficient Relational Treatment using Conceptual Graphs

The straightforward approach in IR systems based on *CGs* is to apply the projection operator from the query graph in each of the document indexes, and retrieve documents whose indexes contain a projection. The iterative application of the operator diminishes the efficiency of the system, because the operator itself is already expensive [5, 20], so applying it for each document in the collection results in high execution time.

In our context, adding another layer to the projection operator to treat relation properties would make things even worse. Our solution to this problem is to decompose conceptual graphs into smaller parts on the relation level, operation whose advantage is threefold. Firstly, the matching function is not applied from the query in each index, but once, from the query in all the organized indexes. Secondly, many operations involved in the projection operator are not performed during retrieval but during indexing, thus giving a faster matching function. Thirdly, derivations on relations are not part of the matching function; instead, they are performed during indexing. The resulting indexes are then implicitly used by the matching function, and relation properties are thus considered without lowering retrieval performance.

Figure 7: The concepts of the graph $\chi(d)$ can be associated with unique labels. The graph can be then decomposed into relations and reorganized into an inverted file structure.

Our approach is motivated by the remark that a conceptual graph contains a set of relations, together with the concepts they link. We need to get to the relation level, in order to deal with relation properties [13]. However, a conceptual graph cannot be simply transformed into the set of its component relations, due to the loss of the graph structure in the process. To preserve it, it is necessary to keep track of the position of relations and concepts in the graph, so we associate a unique label to each of the concepts [5, 21]. The resulting set of relations can be dealt with independently, organized into an efficient structure, and modified using the defined relation properties.

For the graph $\chi(d)$ in figure 3, we associate a unique label to each of its concepts, as shown in figure 7. A label is denoted by a document identifier d_i , unique within the document collection, and an integer identifier $[j]$ which is unique within the document. This operation is based on a logical interpretation for *CGs* that preserves the semantics of the formalism [5]. Then, from the labeled graph, we obtain a set of four relations, that we organize into an inverted

file structure as shown in the same figure. We note that from the labeled $\chi(d)$ one can obtain the inverted file and vice-versa; for instance, the list of labels associated to [BACK OF CHAIR] and [CHAIR] identify precisely the position of the relation [BACK OF CHAIR] $\rightarrow$ (Part-of) $\rightarrow$ [CHAIR]. The occurrence of the same label in the structure signifies a join on the corresponding concepts in the graph. In the general case, the lists of labels in the structure would have more than one element, because all the indexes are grouped into a single structure and occurrences of the same relation linking the same concepts are grouped together (for more details, see [21]).

With this structure, derivations on relation properties can now be performed during indexing. The results of these derivations are only reflected on modifications of the inverted file and not of the indexes. Therefore, whenever it is necessary, the actions performed by derivations can be undone and the inverted file can be rebuilt from the indexes.

For every derivation rule that we have, there is a corresponding treatment on the inverted file, that comes down to the application of a certain algorithm ¹. For instance, the treatment of transitivity is performed by algorithm 1.

Derivations for the inversion property are introduced by a similar algorithm (see algorithm 2). As inversion is a reflexive mathematical property, if the inverse of a relation R_1 is a relation R_2 , then the reverse also holds. Therefore, algorithm 2 is applied twice, once with the input parameters R_1 and R_2 and once with the swapped input parameters, R_2 and R_1 .

Algorithm 1 (Treatment of transitivity)

input: relation type R

FOR each two different relations Rel₁ and Rel₂ of the inverted file

if relation type R in both relations then

Labels₁ = labels for the second concept, in Rel₁

Labels₂ = labels for the first concept, in Rel₂

FOR l in Labels₁ $\cap$ Labels₂

NewC₁ = first concept in Rel₁

NewC₂ = second concept in Rel₂

NewL₁ = label corresponding to l, for the first concept in Rel₁

NewL₂ = label corresponding to l, for the second concept in Rel₂

E = find relation with relation type R, first concept NewC₁ and second concept NewC₂, in the inverted file

if E==nil then

E = new relation with relation type R, first concept NewC₁ and second concept NewC₂

Add E to the inverted file

Add the pair NewL₁,NewL₂ to E, if it is not already in E

ENDFOR

ENDFOR

¹In the algorithms presented here, binary relations are considered. The extension of the algorithms for n-ary relations is straightforward.

Algorithm 2 (Treatment of inversion)

input: relation types R_1, R_2

FOR each relation Rel of the inverted file

 if relation type R_1 in Rel then

 Labels₁ = labels for the first concept, in Rel

 Labels₂ = labels for the second concept, in Rel

 Length = length of Labels₁

 FOR $i = 1$ to Length

 NewC₁ = second concept in Rel

 NewC₂ = first concept in Rel

 NewL₁ = i^{th} label in Labels₂

 NewL₂ = i^{th} label in Labels₁

 E = find relation with relation type R_2 , first concept NewC₁ and second concept NewC₂, in the inverted file

 if E==nil then

 E = new relation with relation type R_2 , first concept NewC₁ and second concept NewC₂

 Add E to the inverted file

 Add the pair NewL₁,NewL₂ to E, if it is not already in E

 ENDFOR

ENDFOR

These algorithms are applied successively to the inverted file, until no more additions can be made. It is proven in [5] that these algorithms terminate; the proof comes out from the characteristics of the conceptual graphs as a relational indexing language. Each addition corresponds to the addition of a relation in the labeled graph. As an illustration, considering the property of transitivity for relation Left-to is equivalent to the application of the algorithm for transitivity, for $R = \text{Left-to}$, on the inverted file in figure 7. The resulting structure in figure 8 corresponds to the addition of a new Left-to relation to the graph.

We can also apply the inversion algorithm, for the relations Left-to and Right-to, which are one the inverse of the other. We extend the inverted file further, by adding a Right-to relation for each Left-to in the inverted file. The index and the inverted file in figure 9 are now larger than the original ones in figure 7. Choosing one or the other inverted file for retrieval makes a big difference on retrieval effectiveness. For a query [Table] $\rightarrow$ (Right-to) $\rightarrow$ [Man: Gabriel Faure], there is no projection in the non-extended labeled graph in figure 7. On the contrary, relational extensions permit to find a projection in figure 9. We note that the inverted file structure can also be used for queries of the kind [Table] $\rightarrow$ (Right-to) $\rightarrow$ [Human], that is for projections involving restrictions on the concepts. To this effect, additional structures are used, that pre-compute these restrictions for any relation that could occur in the query. For details on the projection algorithm on top of the inverted file, see [5, 21].

A) Extended labeled graph obtained for $\chi(d)$

B) Corresponding extended inverted file

Figure 8: Extension on the transitivity of relation Left-to

 A) Extended labeled graph obtained for $\chi(d)$

B) Corresponding extended inverted file

Figure 9: Additional extension on the inversion property between relations Left-to and Right-to

7 Results of the Approach

The arguments that we presented in the previous sections, in support of the treatment of relations using conceptual graphs, as part of the indexing, were confirmed by the improvements on effectiveness and efficiency obtained in our experiments. We implemented the approach into an information retrieval system based on conceptual graphs called RELIEF [22]. We launched thirty queries on a 650 image collection indexed in the form of conceptual graphs. The queries were run twice, once on the inverted file structure built from the original indexes, and once on the extended inverted file, after the relation properties were taken into account.

Figure 10: Relational treatment improves retrieval quality

Retrieval improvement is important in terms of precision /recall, as shown in figure 10. This is due to the fact that extending the inverted file structure corresponds to the addition of new relations to the indexes. Queries that include these added relations may retrieve the corresponding images from the extended file, and not from the original inverted file. We get about 10% of overall improvement, which is an important result [5].

Effectiveness improvement is not achieved at the expense of retrieval efficiency. On the contrary, index reorganization into the inverted file structure, together with a set of structures that essentially pre-compute concept restrictions during indexing (see [5, 21] for details), cut down execution time when compared to previous straightforward or refined applications of the projection algorithm [9].

The longest retrieval times, of 3 seconds, were obtained for queries with relatively general concept types, that is concept types that are placed close to the concept type $\top$. The least execution time was less than 1 second. The average time of almost 2 seconds proves that our approach is workable and that considering relations during indexing has practically no effect on retrieval efficiency, while improving effectiveness. Moreover, as the straightforward application² of the projection operator gives about 17 seconds on average, for the same image collection, queries are processed with our approach about 10 times faster.

²Straightforward application of the projection operator consists in incrementally searching a projection of the query graph in each of the indexes.

8 Conclusion

This paper is an application and a confirmation of the importance of relations and relation properties for accurate indexing and effective retrieval. We present relational extensions to conceptual graphs, according to the general relational indexing framework in [13]. These extensions are sound and are performed by applying different algorithms for the derivations involved in each relation property extension.

Our important concern is to keep retrieval time within reasonable values. We want to obtain a workable system, as it is necessary in the information retrieval domain, where time is a valuable user resource. To this effect, we perform relation-based reasoning as part of indexing, so that not to overload the already complex matching function provided by conceptual graphs. In addition, we apply classical information retrieval techniques in order to cut down execution time. Queries are processed against an inverted file structure, which improves retrieval efficiency.

The experimentations performed on an image collection are encouraging. We achieve effective and efficient query processing. Effectiveness is obtained thanks to the consideration of relation properties in the indexes, whose extension improve the precision/recall values. It is a practical proof of the impact of relational indexing on retrieval effectiveness.

We think that our results are encouraging from a larger perspective. The work gives practical proofs for the impact of relational indexing, and its originality comes from the reasonable retrieval time performance. Similar solutions could be envisaged and applied to other formalisms treating relations, with benefits from both speed and expressiveness points of view.

References

- [1] E.W. Brown. Fast evaluation of structured queries for information retrieval. In *Proceedings the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, USA*, pages 30–38, July 1995.
- [2] Y. Chiamarella and J.P. Chevallet. About retrieval models and logic. *The Computer Journal*, 35(3), 1992.
- [3] C. Meghini, F. Sebastiani, U. Straccia, and C. Thanos. A model of information retrieval based on a terminological logic. In R. Korfhage, E. Rassmussen, and P. Willit, editors, *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pittsburgh, PA*, pages 298–307. ACM, ACM Press, June 1993.
- [4] M. Lalmas. *Theories of Information and Uncertainty for the modelling of Information Retrieval: an application of Situation Theory and Dempster-Shafer's Theory of Evidence*. PhD thesis, Department of Computing Science, University of Glasgow, Scotland, April 1996.
- [5] I. Ounis. *Un modèle d'indexation relationnel pour les graphes conceptuels fondé sur une interprétation logique*. PhD thesis, Université Joseph Fourier, Grenoble, February 1998.
- [6] P. Martin and L. Alpay. Conceptual structures and structured documents. In P.W. Eklund, G. Ellis, and G. Mann, editors, *Proceedings of the 4th International Conference on Conceptual Structures, ICCS'96*, volume 1115 of *Lecture Notes in Artificial Intelligence*, pages 145–159, Sydney, August 1996. Springer-Verlag, Berlin.
- [7] Y. Chiamarella and A. Kheirbek. An integrated model for hypermedia and information retrieval. In *Information Retrieval and Hypertext*. M. Agosti and A. Smeaton (Ed), Kluwer Academic (Publ), 1996.
- [8] D. Genest and M. Chein. An experiment in document retrieval using conceptual graphs. In *Proceedings of the 5th International Conference on Conceptual Structures*, volume 1257 of *Lecture Notes in AI*, pages 489–504, Seattle, USA, August 1997. Springer Verlag.

- [9] M. Mechkour. *EMIR2. Un Modèle étendu de représentation et de correspondance d'images pour la recherche d'informations. Application à un corpus d'image historiques*. PhD thesis, Université Joseph Fourier, Grenoble, 1995.
- [10] M. Mechkour, P. Mulhem, F. Fourel, and C. Berrut. Prime-gc. a medical information retrieval prototype on the web. In *RIDE'97. Seventh International Workshop on Research Issues in Data Engineering: High performance Database Management for Large Scale Applications.*, Birmingham, England, April 7–8 1997.
- [11] Y. Chiaramella, F. Fourel, and P. Mulhem. Modelling multimedia structured documents. Technical report, FERMI BRA 8134, April 1996.
- [12] F. Paradis. *Un modèle d'indexation pour les documents textuels structurés*. PhD thesis, Université Joseph Fourier, Novembre 1996.
- [13] I. Ounis and T.W.C. Huibers. A logical relational approach for information retrieval indexing. In *19th Annual BCS-IRSG Colloquium on IR Research, Aberdeen, Scotland*. EWIC, Springer-Verlag, 8–9 April 1997.
- [14] J.F. Sowa. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley Publishing Company, 1984.
- [15] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill Book Company, New York, 1983.
- [16] T.W.C. Huibers, I. Ounis, and J.P. Chevallet. Conceptual graphs aboutness. In P.W. Eklund, G. Ellis, and G. Mann, editors, *Proceedings of the 4th International Conference on Conceptual Structures, ICCS'96*, volume 1115 of *Lecture Notes in Artificial Intelligence*, pages 130–144, Sydney, August 1996. Springer-Verlag, Berlin.
- [17] C. J. van Rijsbergen. A new theoretical framework for information retrieval. In *ACM Conference on Research and development in Information Retrieval, Pisa*, pages 194–200, 1986.
- [18] M.L. Mugnier. *Contributions Algorithmiques pour les Graphes d'Héritage et les Graphes Conceptuels*. PhD thesis, Université Montpellier II, October 1993.
- [19] E. Salvat and M.L. Mugnier. Sound and complete forward and backward chainings of graph rules. volume 1115 of *Lecture Notes in Artificial Intelligence, Subseries of Lecture Notes in Computer Science*, pages 248–262, Sydney, Australia, August 1996. Springer-Verlag.
- [20] M. Chein and M.L. Mugnier. Conceptual graphs: fundamental notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.
- [21] I. Ounis and M. Pasca. A promising retrieval algorithm for systems based on the conceptual graphs formalism. In *International Database Engineering & Applications Symposium (IDEAS'98)*, Cardiff, UK. IEEE Computer Society Press, July 1998.
- [22] I. Ounis and M. Pasca. The RELIEF retrieval system. In *The IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'97)*, Newport Beach, California, U.S.A. IEEE Computer Society Press, November 1997.