RELIEF: Combining expressiveness and rapidity into a single system

Iadh Ounis
CLIPS-IMAG
Université de Grenoble, France
www-clips.imag.fr/mrim/iadh.ounis/

Marius Paşca
CLIPS-IMAG
Université de Grenoble, France
www-clips.imag.fr/mrim/marius.pasca/

Abstract This paper constitutes a proposal for an efficient and effective logical information retrieval system. Following a relational indexing approach, which is in our opinion a necessity to cope with the emerging applications such as those based on multimedia, we use the conceptual graphs formalism as our indexing language. This choice allows for relational indexing support and captures all the useful properties of the logical information retrieval model, in a workable system. First order logic and standard information retrieval techniques are combined together, to the same effect: obtaining an expressive system, able to accurately handle complex documents, improve retrieval effectiveness, and achieve good time performance. Experimentations on an image test collection, within a system available on the Web, provide an illustration of the role that logic may have in the future development of information retrieval systems.

1 Introduction

The emergence of new applications, such as those based on multimedia, brings into discussion the problem of the accuracy of document representation in information retrieval (IR). Highly structured documents require more complex indexing languages [5, 13, 9]. Following the work of Farradane [7], we think that relational indexing is a necessity if faithful representations of documents are to be obtained. The complexity of a multimedia document, such as the temporal aspects related to video or the geographical position of objects in an image, suggests that keywords are not sufficiently expressive and the *relation* becomes highly important, and cannot be overlooked in the indexing process. Recent work showed that, when relations are taken into account, retrieval effectiveness can be improved [19, 20, 18].

Representing relations between keywords is a more difficult task, as we deal in this case with a more refined level of information. Moreover, once relational indexing is supported by the system, a powerful matching function is needed in order to fully exploit the knowledge captured by or associated to these relations. We think that the logical IR model [27] is the only model suitable for this task. The system must be able to manage complex indexes and take benefit from large quantities of knowledge, in order to improve the quality of the retrieval process. In our opinion, retrieval should be a deductive process, as only logic provides sound, rule-based knowledge matching.

An expressive indexing language, which complies with the need of relational indexing, introduces more complex treatments as part of the matching function. Indeed, in [10, 11] it was shown that the use of expressive formalisms typically implies more expensive treatments. As in IR the acceptance of a system by its users greatly depends on its ability to provide a fast matching function [2], there were some doubts related to the practicability of a logical IR system based on complex relational formalisms. More elaborate implementations seemed to confirm that in this case it is difficult to achieve good time performance [16, 17].

This paper is a challenge, in that we argue that the logical IR model is practicable and it is a self-contained model, with the advantages given by flexibility and power of deduction mechanisms. To achieve a practicable implementation while using a relational indexing, we use conceptual graphs [26]. We believe that this knowledge representation formalism has net advantages in information retrieval, such as the ability to model all the components of an IR system, the existence of a powerful logical matching function, and easy extensions without changes of the semantics of the formalism. The latter make it possible to render the matching function even more powerful, by refining the retrieval decision [8] on the basis of knowledge such as that captured by relation properties [22].

We use first order logic and standard information retrieval techniques to achieve important improvements in retrieval time performance. The approach is applied on an image retrieval system based on conceptual graphs. For an image collection of 650 images, experimental evidence confirms the soundness of the theoretical basis and opens a new path towards a fast precision-oriented retrieval system.

The main contributions of the paper are the following:

- we illustrate the role of relations in image indexing and show how the image content can be captured by conceptual graphs.
- we show that the matching function between queries and image indexes can be performed in polynomial time, if graphs are considered from a logical point of view, as first order logic expressions.
- we show how to retrieve items from an image collection. We use an inverted file approach to store and retrieve document information, while preserving the deductive power of first order logic. Deduction is thus seen as a process of retrieval, in which we can simultaneously deduce all relevant information from the document collection.
- we give experimental evidence. We show that conceptual graphs are expressive enough to model multimedia information. Retrieval effectiveness is improved, while time performance is very promising.

2 Representing Images by Conceptual Graphs

We consider the two images in figure 1 to review some of the types of information specific to this media and important for in-

dexing. Both images contain objects, that may be identified and represented at first view by the keywords MAN and TABLE. Each man has a BEARD. For simplicity, we ignore the other objects that appear in the images. In this case, if we use keywords, both images will be indexed by the same set of objects: {MAN, TABLE, BEARD}. Therefore, whatever the user's query, both will be either retrieved or not.

Nevertheless, the two images contain further information that permits to refine their description. For example, our objects have different relative positions. In the first image, the MAN is on the right of the TABLE. The same relation also holds for the second image. Moreover, the BEARD does not appear independently in each image, but it is the MAN which has it. The position of the MAN also makes the difference: he is seated in the first image, but is standing in the second image. We would also like to introduce the name of the photographers who took the two images, as well as the identification of the photographed persons. In the database community, such information is usually referred to as attributes.

Keywords do not provide a solution to our needs. If we tried to use a brute-force method and organized some of the above information in the form of keywords, we could get the sets {MAN, TABLE, BEARD, ON THE RIGHT, SEATED} and {MAN, TABLE, BEARD, ON THE RIGHT, STANDING} respectively. Still, what we obtain does not make much sense, and one could only make guesses at whether it is the man which is on the right of the table, or maybe the table is on the right of the man etc. This stems from the fact that relations between keywords cannot be modeled by keywords themselves.

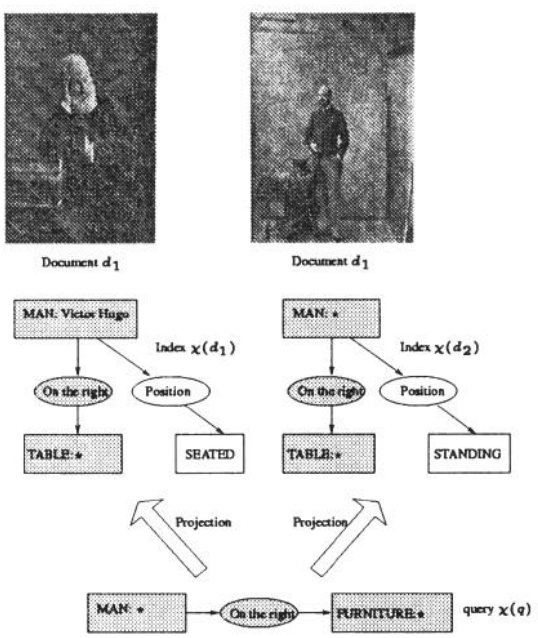

Figure 1: Two images with their associated indexes. The query $\chi(q)$ can be projected in both indexes.

We need a formalism that supports relations, which offers a precise relational indexing process. Among other possible choices, we choose Sowa's conceptual graphs (*CGs*) [26] as our expressive indexing language. The reason is that this formalism allows to represent uniformly all the components of an IR system, i.e. indexing, interrogation and matching function [8]. Furthermore, as introduced by Sowa, conceptual graphs can be easily extended to accommodate specific knowledge and needs, without a revision of its semantics. [1]

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 [26]. Concept nodes represent entities, attributes, states and events, and relation nodes show how the concepts are interconnected. A concept has a *type* (which corresponds to a semantic class) and possibly a *marker* (which corresponds to an instantiation to an individual of the class). For instance, [MAN : $*$] stands for the concept of all possible men. This concept is called a *generic* concept also noted [MAN]. On the other hand, [MAN : Victor Hugo] obviously stands for the concept of a man named Victor Hugo.

We note that $\star$ and Victor Hugo are examples of the notion of marker. More formally, there is a set $\mathcal{M} = \zeta \cup W \cup \{\star\}$ of *markers* for individuals belonging to the concept type domains. The extra sets ζ and W are used to denote individuals in a concept type domain. $\zeta = \{John, Mary, \dots\}$ is called the set of individual markers. ζ and W are disjoint sets. The use of the set W, called the set of *witnesses*, will be explained in Section 4.

We use a *conformity relation* between concept types of $\mathcal{T}_c$ and markers of $\mathcal{M}$. In particular, for every element $d \in \zeta$ or $w \in W$ and for every concept type $C \in \mathcal{T}_c$ the conformity relation tells whether $C(d)$ and $C(w)$ can be interpreted true.

A relation has only a *type*. Concept types and relation types are organized in different lattices, $\mathcal{T}_c$ and $\mathcal{T}_r$ respectively. These lattices are built according to a partial ordering relation $\leq$ between types. The number of concepts linked by a relation must be equal to the arity of its relation type. Each relation type r is semantically defined by a relation signature, $\Sigma_r = (r, n, C_1, \dots, C_n)$, where n is the arity of the relation type r and $C_1, \dots, C_n$ are the greatest concept types in the lattice $\mathcal{T}_c$ which can be linked by the relation type r *in this order*. Relation signatures impose a boundary upon the argument concepts. Only specializations of these concepts can appear as arguments of relations in documents and queries. For example, if we have $\Sigma_{\text{On the right}}$=(On the right, 2, PERSON, TABLE), it is not possible that we have the subgraph [PERSON]→(On the right)→[FURNITURE] in any of the documents, nor in the queries because TABLE $\leq$ FURNITURE.

Given a document collection, we have a conceptual graph that indexes each document. For a document d_i, i $\in$ [1,document collection size], we denote its index 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 [FURNITURE] by [TABLE]), or on the marker, by instantiating the generic marker $\star$ to a certain individual marker (for instance, replacing [MAN] by [MAN: Victor Hugo]), or on both (see Document d_1 in figure 1).

An illustration of the use of projection as the matching function is given in figure 1. In this case, we have a projection of the query $\chi(q)$ in the indexes $\chi(d_1)$ and $\chi(d_2)$. For instance, in the case of $\chi(d_1)$, the result of the projection is the darkened subgraph of $\chi(d_1)$, and it is obtained by restricting respectively the concept [MAN] in [MAN: Victor Hugo] and the concept [FURNITURE] in [TABLE].

One of the drawbacks of the projection operator is that it does not take into account general domain knowledge in the graph matching. Such domain knowledge can be very useful,

[1] This is opposed to, for example, the case of the terminological logics [23], where extensions generally induce the introduction of new semantic interpretations.

and in relational indexing it may include relation properties, such as symmetry, transitivity, inversion etc. The relation properties are all captured in a set $\mathcal{K}_s$, that we take into consideration to refine the indexes and thus improve retrieval effectiveness [22]. An illustration of the use of $\mathcal{K}_s$ is given in section 7.

Conceptual graphs can be used in the context of van Rijsbergen's logical IR model introduced in [27]. 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.

In *CGs*, the implication is obtained on the basis of the translation to first-order logic, by the ϕ operator given by Sowa with his formalism [26]. 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, $\phi(h) \supset \phi(g)$ [26, 14].

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 [8]. 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 1 from $\chi(q)$ in $\chi(d_1)$, we also have an implication in terms of logical formulas, in the opposite direction, $\phi(\chi(d_1)) \supset \phi(\chi(q))$. In the following, we shall focus on the projection operator.

3 Motivations

In [4, 15], it is argued that the projection of a conceptual graph in another is an operation that cannot be performed in polynomial time. Brute force implementations of the projection in a *CG*-based IR system would result in unacceptable execution times. Therefore our goal is to resolve this problem in the context of IR, such as the semantics of the formalism be preserved and rapidity be achieved as well.

An IR system has two main functionalities: indexing and retrieval. When globally evaluating the time performance of the system, the retrieval procedure is the essential parameter to be taken into account. This suggested our primary solution to reduce execution time. For our projection operator, it would be advantageous to move as much of the complex treatments as possible from retrieval to indexing. We organize conceptual graphs adequately, and here we were inspired by classical techniques in information retrieval, such as the use of the inverted files [28, 25], which are common practice in commercial IR systems. When keyword-based formalisms are used for indexing, the inverted file consists of entries that associate each indexed keyword to the set of documents whose index contain it. This technique improves speed, as it provides directly the documents corresponding to each indexing term. Hence our idea to apply it to a rich formalism such as conceptual graphs. We obtain a global structure, which is built during indexing. Treatments that are otherwise part of the projection operator are performed during indexing and their results are stored in the global structure. We prove in the following section that our particular organization of *CG*s does not change the semantics of the formalism. In particular, the result of the application of our matching function is equivalent to that of the projection operator. The difference is that we obtain it faster, as less operations remain to be performed during retrieval.

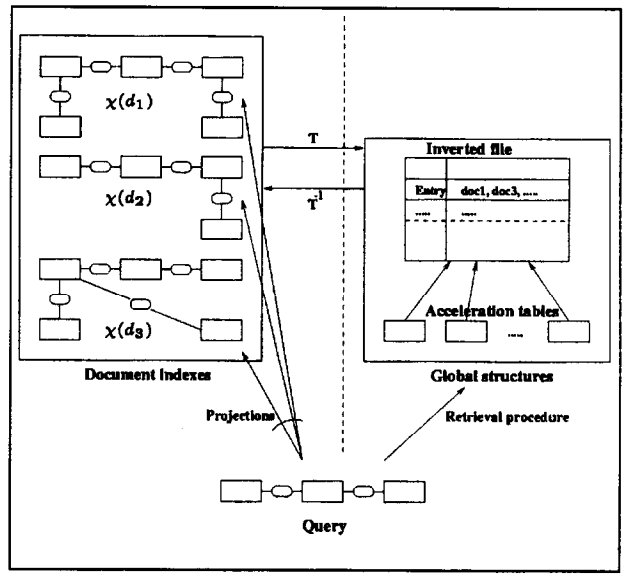

Figure 2: Comparative views of the classical projection as a matching function (on the left) and of our approach (on the right)

The main idea is illustrated in figure 2. The left side of the figure corresponds to the brute force use of the projection operator. During retrieval, the algorithm corresponding to the projection operator has to be sequentially applied between the query and each of the document indexes. On the right side, our approach is presented. Using a transformation T, the indexes are organized in a global structure. This structure contains an inverted file and a set of acceleration tables that store pre-computed data that are needed to perform the projection algorithm, in particular all possible specializations of a query subgraph (we shall come back to their role later). It should be pointed out that the transformation process is bijective. Therefore it is a lossless process, and each index can be built back from the structure. This is explained by the underlying logical interpretation.

In the following, we firstly present the theoretical logical support that allows to organize conceptual graphs in inverted files. Then we show the transformation of the initial problem, of the projection of one graph in another. This allows to organize the indexes of the whole document collection in a single structure. It gives the particular advantage that the matching is not performed locally, between the query and each conceptual graph, but globally, as a single operation on our structures.

4 The Witness Technique

Before we introduce the notion of witness, let us recall the definition of the formula operator ϕ introduced by Sowa [26]. For the sake of simplicity only dyadic relations are used to illustrate how this operator works, but it is easy to generalize it to an arbitrary n-ary relation. Let us take an elementary piece of a conceptual graph that we call *an arch*; that is, $[C_1 : \star] \rightarrow (Rel) \rightarrow [C_2 : \star]$ where C_1 and C_2 are concept types and Rel is a dyadic relation which holds between them. From the relation signature Σ_{Rel} we know which concept of the two has to be considered in the first place in the relation Rel. The ϕ operator gives an existential interpretation to the marker $\star$:

$$\phi([C_1 : \star] \rightarrow (Rel) \rightarrow [C_2 : \star]) = \exists x_1 \exists x_2 (C_1(x_1) \wedge C_2(x_2) \wedge Rel(x_1, x_2)) \quad (1)$$

The logical interpretation $\phi(g)$ of a conceptual graph g is the existential closure of the conjunction of the formulae associated with its *arches*. Hence, if a concept node relates two or more arches in a graph, like the concept $[C_{12}]$ of $\chi(q)$ in figure 4, then the translation of the associated subgraph in $\chi(q)$ is

$$\exists x_1 \exists x_2 \exists x_3 (C_1(x_1) \wedge C_{12}(x_2) \wedge R_2(x_1, x_2) \wedge C_1(x_3) \wedge C_{12}(x_2) \wedge R_1(x_3, x_2))$$

Now, the method of the constants [3] says that we can introduce for each existential quantifier a new constant $w \in W$ which witnesses the truth of the existential formula. For example the above translation can be considered as equivalent to

$$C_1(w_1) \wedge C_{12}(w_2) \wedge R_2(w_1, w_2) \wedge C_1(w_3) \wedge C_{12}(w_2) \wedge R_1(w_3, w_2)$$

The equivalence holds only if we add the extra axiom $\exists x\ \alpha(x) \rightarrow \alpha(w)$ to the standard translation of the ϕ operator, provided that w is a *new* constant and it has been substituted for all free occurrences of x in the formula α (notice that also the converse implication holds). If a formula without witnesses can be derived from the extended translation, then it can be also obtained from the standard translation (notice that also the inverse direction trivially holds). In particular

Theorem 1 *If the logical counterpart of a query not containing witnesses can be derived by the extended graph translation, then it can be obtained from Sowa's graph translation and vice-versa.*

The proof can be found in [18]. This theorem represents the theoretical basis of our approach. It insures that, whenever a graph can be projected in another graph according to Sowa's projection operator, it is also projected according to our witness-based interpretation. The latter is indeed sound and complete according to Sowa's interpretation.

We note that the witness technique is applied not only to existential concepts but also to individual concepts. In both cases, a unique witness is associated to each concept node of the graph.

5 Transforming the Projection Operator into Two Sub-problems

After introducing in section 4 the logical foundation, we shall concentrate on the organization of the indexes (the conceptual graphs) for faster retrieval. Our structure consists of an inverted file and several acceleration tables. The inverted file permits to treat the indexes globally. The acceleration tables permit to focus the retrieval from the beginning, reducing the search space and thus improving efficiency. The construction of the inverted file and the acceleration table is done off-line, as part of the indexing procedure.

An uniform treatment of the arches in all the indexes is achieved by the assignation of a unique witness to each concept, as mentioned in the previous section. Provided that the witnesses can be distinguished from one another , we can treat the arches separately, and we end up with a set of arches for all the collection. Witnesses will allow for the reconstruction of any of the original indexes. For witnesses, we shall use the notation w_i^j, where i is the unique identifier of the document and j is a unique identifier within i. Thus the witnesses obtained remain distinct for all the indexes.

It is from this set of arches that the construction of our structures is done. During retrieval, the inverted file will be used to find directly where each arch of the query appears in the indexes. We group in the same entry of the inverted file all the arches in the collection which are syntactically equal. This means that, given an arch, we can immediately locate the documents whose indexes contain that arch. This is the basis for the construction of the inverted file.

The acceleration tables pre-compute all the specializations of possible query arches before interrogation. For each arch of the query graph, all the entries of the inverted file containing a specialization of this arch will be quickly determined. Then the documents that contain specializations of each of the query arches are found by consulting the inverted file entries.

At this point, we divided the projection operator into two sub-problems. One of them is dealt with during indexing, hence it does not have any influence on the retrieval rapidity. It results in modifications on the inverted file and the acceleration tables. The other sub-problem remains to be solved during retrieval and, together with the first problem, it accomplishes the task of the projection operator. The retrieval procedure takes benefit from the pre-computed data, therefore is faster. This is represented by the following formula:

Projection = Transformation + Pre-computation (specializations) + Retrieval procedure

An important part of the complexity of the projection operator is moved at indexing time (the transformation of graphs into arches and the pre-computation of the specializations of query arches). What it remains is an operation that can be performed much faster, as we shall show in the following.

6 Retrieval

By making use of the pre-computations that are comprised in the global structures (specifically, the acceleration tables), the retrieval function performs in one operation the rest of the calculus to arrive at a result equivalent to that of Sowa's projection (see theorem 1).

We define a joining witness as a witness assigned to a joined concept, which is a concept that appears in more than one arch. The retrieval strategy is designed according to the structure of the query. For each joining witness Wit_q^i, we find, using the acceleration tables, the specializations of each arch of the query in which Wit_q^i appears. For each of these arches, we extract from the inverted file the lists of witnesses corresponding to the position of Wit_q^i in the query arch [2]. We do the intersection of these lists. Thus for each joining witness of the query we have a list of witnesses that gives the documents which are candidate for a possible projection. Hence, for each Wit_q^i we obtain the subgraphs in the document indexes corresponding to the projection of a part of the query (this part consists of the arches of the query in which Wit_q^i appears).

The procedure is illustrated in figure 3. For clarity, we only represent *CGs* by witnesses. The joining witnesses in the query are Wit_q^1 and Wit_q^2. Wit_q^1 appears in arches A and B. Thus we intersect A_d and B_d and we obtain the candidate set of witnesses for Wit_q^1. Analogously, for Wit_q^2 we get $B_d \cap C_d$.

The candidate witnesses allow to detect partial projections. They have to be filtered in order to find the result of the projection. The operations done up to now gave us projections for all subgraphs of arches connected on the same concept (that is, on the same joining witness). Each witness in $A_d \cap B_d$ and each witness in $C_d \cap D_d$ are checked, to verify that they belong to the same relation that joins Wit_q^1 and Wit_q^2. The pairs that verify

[2] The position is determined by the relation signature.

this condition are retained and then assembled as the result of the projection (see algorithm below). The corresponding projections can be found from the inverted file, by means of the inverse transformation T^{-1}.

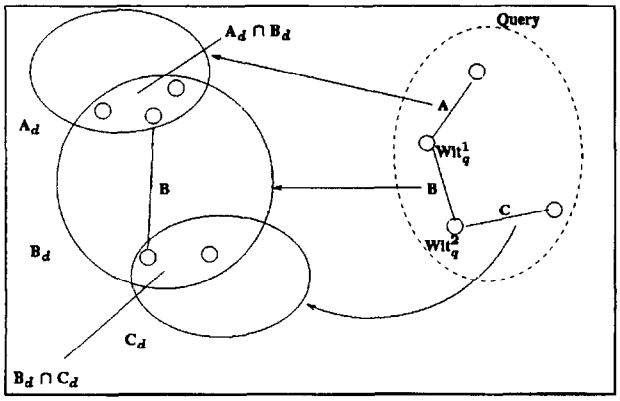

Figure 3: The retrieval procedure

Algorithm 1 (Retrieval)

1) FOR each joined Wit_q^i *of the query*
  $JoinedArches_q$ = FINDJOINEDARCHES(Wit_q^i)
  $PartialProj_{Arch_q}$ = *all witnesses*
  2) FOR each $Arch_q$ *of* $JoinedArches_q$
    $SpecializedRowArch_q$ = FINDSPECIALIZATIONS($Arch_q$)
    $AllSpecializedWit = \emptyset$
    3) FOR each $RowArch_q$ *in* $SpecializedRowArch_q$
      $SpecializedWit_{RowArch_q}$ = GETWITNESSES($RowArch_q$)
      $AllSpecializedWit = SpecializedWit_{RowArch_q} \cup AllSpecializedWit$
    ENDFOR
    $PartialProj_{Arch_q} = PartialProj_{Arch_q} \cap AllSpecializedWit$
  ENDFOR
ENDFOR
$PartialProj_q$ = FINDALLPARTIALPROJECTIONS($PartialProj_{Arch_q}$)
$ExactProjection_q$ = FINDALLPATHSBETWEENWIT$_q$ ($PartialProj_q$)

The use of the inverted file and the acceleration tables are explicit in the algorithm, as opposed to the illustration in figure 3. Its main operations were illustrated above. A few details about the symbols used in the algorithm above are given here:

FINDJOINEDARCHES(Wit_q^i): function to find all the arches of the query in which Wit_q^i appears.

FINDSPECIALIZATIONS($Arch_q$): function to find, in the acceleration table of the relation of $Arch_q$, the inverted file entries whose arches are specializations of $Arch_q$.

GETWITNESSES($RowArch_q$): function that gets the witnesses associated to a concept of the arch given in the $RowArch_q$.

FINDALLPARTIALPROJECTIONS($PartialProj_{Arch_q}$): function that extracts the document identifiers of the witnesses corresponding to the set $PartialProj_{Arch_q}$. It finds from the precedent results all the possible partial projections in the document indexes of the whole query.

FINDALLPATHSBETWEENWIT$_q$($PartialProj_q$): function that verifies that the obtained witnesses are linked together in the documents, according to the query structure. It is used to find the final relevant documents, and thus the exact projection of the query as a whole on the documents. Its sub-steps were described above.

In the retrieval algorithm, the partial projections are obtained in polynomial time, as the first part of the algorithm consists in unions and intersections. The overall complexity is therefore given by the complexity of the last function. In the conceptual graphs community the cost of this function is over-estimated, as it is often linked to graph theory and considers graphs of indefinite structure. However, we consider a conceptual graph to be a logic formula, hence the graphs are always normalized, to eliminate expressions like C(a)∧C(a) and replace them by C(a). Considering *CGs* as a logic allows to have a polynomial complexity because, for normalized graphs, a subgraph of the query can be projected in at most one subgraph of a document index. Hence, according to the query structure, there is only one way to verify that the document index complies with this structure, i.e. to verify that its subgraphs are linked together similarly to the query subgraphs. This shows why the function FINDALLPATHSBETWEENWIT$_q$ is polynomial. Hence, the overall complexity of our algorithm is polynomial. More details about the complexity analysis of the projection are given in [1]. We emphasize that graph normalization is a consequence of a logic view of *CGs*, and it does not impose further constraints; in particular, graphs can be cyclic without any change in the algorithm, nor in its complexity.

7 An Illustration of How the Algorithm Works

Let us consider the example of figure 4. The continuous lines correspond to strict indexes, as provided by the indexing process. The dotted lines are extensions added according to relation properties in $\mathcal{K}_s$. Their addition to the strict indexes will be reflected by modifications in the inverted file and will be taken into account by the retrieval procedure. Let the relation signatures Σ_{R_1},Σ_{R_2} and Σ_{R_3} be: $(R_1, 2, C_1, C_1)$, $(R_2, 2, C_1, C_1)$, $(R_3, 2, C_1, C_1)$. For simplification, we suppose that in the following $C_{ij} \leq C_i$, where i and j are natural numbers (i.e., C_{ij} is a sub-type of C_i in the lattice $\mathcal{T}_c$). The reader may note that in our example, we do not apply any graph normalization; the reason is that we want to illustrate the futility of such a graph of indefinite structure, as well as the application of our retrieval algorithm for all kinds of graphs.

As we mentioned in section 5, the *CGs* can be treated as a set of arches, where each concept is associated with a distinct witness (see figure 4). For practical reasons, we deal with witnesses according to the documents in which they appear and use in the following the notation $d_i[j]$ for the general w_i^j. For example, the set of witnesses of $\chi(d_1)$, $[w_1^1, w_1^2, w_1^3, w_1^4]$, is represented by $d_1 : [1, 2, 3, 4]$.

Building global structures

In the inverted file, the arches are grouped together in the same entry, whenever they are syntactically equal. For each arch, concepts and their corresponding witnesses are represented vertically, according to their position in the arch. This position is determined by the relation type signature (see section 2).

To improve clarity, we shall use in the following figures the notation C for the concept [C:⋆] and $C : a$ for [C:a], where a is an individual marker.

The inverted file corresponding to the strict indexes is shown in figure 5. In each entry the concept represents a pair [concept type, marker]. This is illustrated by the first entry of the inverted file, where the concept type C_1 is followed by the individual marker a. The entry thus corresponds to all the occurrences of the arch $[C_1 : a] \rightarrow (R_1) \rightarrow [C_{11}]$ in the indexes. In the same way, the second entry describes all the occurrences of the arch $[C_{11}] \rightarrow (R_1) \rightarrow [C_{12}]$ amongst the set of arches of all indexes. Thus, $d_1 : [2]$ (w_1^2) and $d_1 : [3]$ (w_1^3) are the witnesses that are attached to the arch in $\chi(d_1)$, while in $\chi(d_2)$ the arch appears for witnesses $d_2 : [5]$ (w_2^5) and $d_2 : [4]$ (w_2^4), in this order (see figure 4).

The inverted file in figure 5 only takes into consideration the strict indexes. It can be extended by using additional knowledge

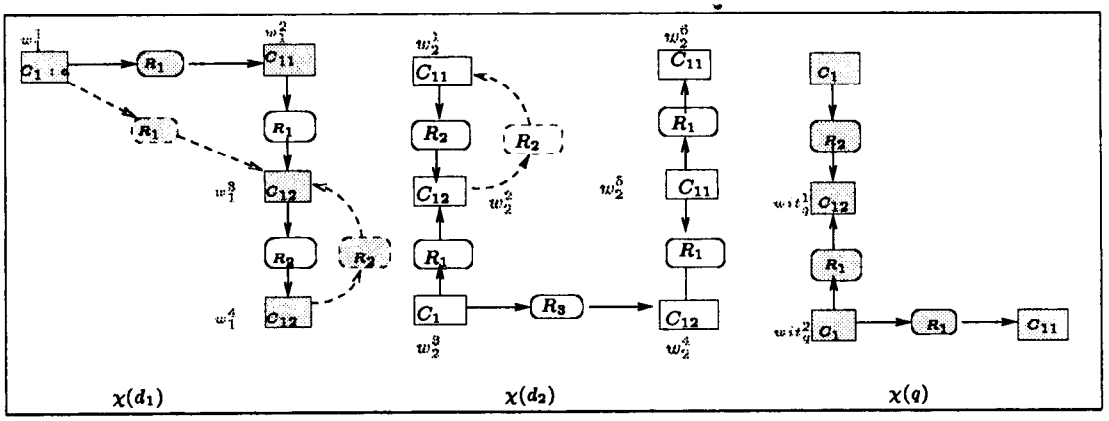

Figure 4: Sample document base and a query that can be projected in the extended index of d_1

about the domain, in our case by using the relational properties of $\mathcal{K}_s$. For every relational property in $\mathcal{K}_s$, there is a corresponding treatment on the inverted file, which reduces to the application of a certain algorithm [22]. For instance, if $\mathcal{K}_s$ contains the facts: R_1 is transitive and R_2 is symmetric, then we obtain the inverted file in figure 6, which will be used to build the acceleration tables and then in the retrieval algorithm.

Row number	*Relation*	*Concepts*	*Witnesses*
1	R_1	$C_1 : a$	$d_1 : 1$
		C_{11}	$d_1 : 2$
2	R_1	C_{11}	$d_1 : 2, d_2 : 5$
		C_{12}	$d_1 : 3, d_2 : 4$
3	R_2	C_{12}	$d_1 : 3$
		C_{12}	$d_1 : 4$
4	R_2	C_{11}	$d_2 : 1$
		C_{12}	$d_2 : 2$
5	R_1	C_1	$d_2 : 3$
		C_{12}	$d_2 : 2$
6	R_3	C_1	$d_2 : 3$
		C_{12}	$d_2 : 4$
7	R_1	C_{11}	$d_2 : 5$
		C_{11}	$d_2 : 6$

Figure 5: Inverted file for our example

Row number	*Relation*	*Concepts*	*Witnesses*
1	R_1	$C_1 : a$	$d_1 : 1$
		C_{11}	$d_1 : 2$
2	R_1	C_{11}	$d_1 : 2, d_2 : 5$
		C_{12}	$d_1 : 3, d_2 : 4$
3	R_2	C_{12}	$d_1 : 3, 4$
		C_{12}	$d_1 : 4, 3$
4	R_2	C_{11}	$d_2 : 1$
		C_{12}	$d_2 : 2$
5	R_1	C_1	$d_2 : 3$
		C_{12}	$d_2 : 2$
6	R_3	C_1	$d_2 : 3$
		C_{12}	$d_2 : 4$
7	R_1	C_{11}	$d_2 : 5$
		C_{11}	$d_2 : 6$
8	R_2	C_{12}	$d_2 : 2$
		C_{11}	$d_2 : 1$
9	R_1	$C_1 : a$	$d_1 : 1$
		C_{12}	$d_1 : 3$

Figure 6: Extended inverted file for our example

Relational properties are not restricted to only symmetry and transitivity. Other properties can be defined and the inverted file would be modified accordingly [22]. However, we shall only consider the inverted file in figure 6.

To build the acceleration tables, the rows of the final inverted file are analyzed in turn. For each relation type, there is one acceleration table. Their content serves to give the specializations for each possible arch that could appear in the query. These possible arches are deduced from the relation signatures. For example, from the relational signatures $\Sigma_{R_1}=(R_1, 2, C_1, C_1)$ we know that the well-formed arches that can appear in the query and are generalizations of $[C_{11}] \rightarrow (R_1) \rightarrow [C_1]$ are $[C_{11}] \rightarrow (R_1) \rightarrow [C_1]$ (the arch itself) and $[C_1] \rightarrow (R_1) \rightarrow [C_1]$. The acceleration table for relation R_1 will have to allow for quick access to the specializations of each of these arches.

The acceleration tables for our example are those of figure 7. We use the notation $a_i(R)$ for the i^{th} concept of the relation R, according to the signature Σ_R. Note that individual markers are also dealt with; the individual marker a that appears in entries 1 and 9 of the inverted file (see figure 6) has an impact on the form of the first two entries of the column $a_1(R_1)$ of figure 7A). This is due to the fact that $[C_1 : a] \leq [C_1]$. The individual markers are also organized in a lattice, so that any individual marker is a specialization of the generic marker $\star$. Therefore the lattice of concept types and the lattice of individual markers are used uniformly to build the acceleration tables.

To illustrate the role of the acceleration tables, suppose that the query contains the arch $[C_1] \rightarrow (R_1) \rightarrow [C_{12}]$. Then from the acceleration table for R_1 (see figure 7A)) we can obtain its specializations; we extract the inverted file entries corresponding to concept C_1 from $a_1(R_1)$ and to concept C_{12} from $a_2(R_1)$ and we intersect them. Thus we have [1,2,5,7,9] $\cap$ [2,5,9] = [2,5,9], so the specializations for our arch, that appear in the indexes, can be obtained by consulting entries 2, 5 and 9 of the inverted file (see figure 6).

A)

$a_1(R_1)$	*Inv. file rows*	$a_2(R_1)$	*Inv. file rows*
C_1	[1,2,5,7,9]	C_1	[1,2,5,7,9]
$C_1 : a$	[1,9]	C_{11}	[1,7]
C_{11}	[2,7]	C_{12}	[2,5,9]

B)

$a_1(R_2)$	*Inv. file rows*	$a_2(R_2)$	*Inv. file rows*
C_1	[3,4,8]	C_1	[3,4,8]
C_{11}	[4]	C_{11}	[8]
C_{12}	[3,8]	C_{12}	[3,4]

C)

$a_1(R_3)$	*Inv. file rows*	$a_2(R_3)$	*Inv. file rows*
C_1	[6]	C_1	[6]
		C_{12}	[6]

Figure 7: Acceleration tables for relations R_1, R_2, R_3

Document Retrieval

To see how the acceleration tables and the inverted file are used for retrieval, suppose that a user enters the query presented in figure 4. There are two joining witnesses in the query. The first one is $q : [1]$, that appears both in $[C_1] \rightarrow (R_2) \rightarrow [C_{12}]$ and in $[C_1] \rightarrow (R_1) \rightarrow [C_{12}]$. The second one is $q : [2]$, that joins $[C_1] \rightarrow (R_1) \rightarrow [C_{12}]$ and $[C_1] \rightarrow (R_1) \rightarrow [C_{11}]$.

For each of the joining witnesses of the query, we have to find its specialized joined witnesses in the document indexes, if such witnesses exist. Thus we project parts of the query in the indexes, using therefore a partial projection; each of these parts is composed of the arches joined by the joining witnesses. Then, to find the projection, the partial results have to be assembled together, to perform the last operation of the retrieval algorithm presented before.

We start with the first joining witness. In the first arch $A = [C_1] \rightarrow (R_2) \rightarrow [C_{12}]$, $q : [1]$ is associated to $[C_{12}]$. From the two columns of the acceleration table for R_2, we find that the arches in which A can be projected are those of the rows [3,4,8] $\cap$ [3,4] = [3,4].

The second arch of the query that is joined by $q : [1]$ is: $B = [C_1] \rightarrow (R_1) \rightarrow [C_{12}]$. In the acceleration table for R_1, we look for $[C_1]$ in the first column and $[C_{12}]$ in the second one. We have [1,2,5,7,9] $\cap$ [2,5,9] = [2,5,9] so there are three entries of the inverted file that contain arches in which $[C_1] \rightarrow (R_1) \rightarrow [C_{12}]$ can be projected.

In order to get all the specializations of A, we do the union of the witnesses lists associated to the second concept in rows 3 and 4 that we have found for arch A of the query: $[d_1 : [4,3]] \cup [d_2 : [2]] = [d_1 : [3,4], d_2 : [2]]$. For the second arch B of the query, our result were rows 2, 5 and 9 of the inverted file; the union of the witnesses list associated to the related concept of those rows is $[d_1 : [3], d_2 : [4]] \cup [d_2 : [2]] \cup [d_1 : [3]] = [d_1 : [3], d_2 : [2,4]]$. Now, only the set of the documents for which there is a join on the concept $[C_{12}]$ associated to wit_q^1 will be a correct answer to the subgraph of the query formed by arches A and B. To find the possible relevant documents, which means the partial projections corresponding to this joining witness, we only have to do the intersection of the last two lists: $[d_1 : [3,4], d_2 : [2]] \cap [d_1 : [3], d_2 : [2,4]] = [d_1 : [3], d_2 : [2]]$. These are the specialized witnesses for the joining witness $q : [1]$, from which the partial projections mentioned above can be found by using the content of the inverted file.

We now turn our attention to the second joining witness of the query, $q : [2]$. It joins arches B, introduced above, and $C = [C_1] \rightarrow (R_1) \rightarrow [C_{11}]$. The same procedure is applied as for $q : [1]$. Therefore the specialized witnesses for $q : [2]$, from which the projections of the subgraph of the query formed by B and C can be derived, are $[d_1 : [1], d_2 : [5]]$.

There is one more operation to perform, in order to find the exact projection of the query in the document indexes and thus find the relevant documents. It corresponds to the function FINDALLPATHSBETWEENWIT$_q$(PartialProj$_q$) of the retrieval algorithm. The two joining witnesses $q : [2]$ and $q : [1]$ of the query (see figure 4) are associated to the concepts of the same arch B, which contains the relation R_1. The specialized witnesses that we found, $[d_1 : [1], d_2 : [5]]$ (for $q : [3]$) and $[d_1 : [3], d_2 : [2]]$ (for $q : [2]$), must fulfill the same condition. It turns out that only $[d_1 : [1]]$ and $[d_1 : [3]]$ do so, while $[d_2 : [5]]$ and $[d_2 : [2]]$ do not belong to a common relation R_1 as required. Therefore, an exact projection from the query exists only in the first document index, and this is the only document that is retrieved by the algorithm.

The interpretation of the result shows that our witness-based retrieval algorithm generates the same results as the classical projection. Moreover, relational reasoning allows to retrieve document d_1, whose non-extended index does not contain an exact projection of the query. We obtain not only the documents that are relevant to the query but also the projection of the query in those documents. To exemplify, in the final result, $[d_1 : [1]]$ and $[d_1 : [3]]$ have the following interpretation: the projection of the query in the index of d_1 exists and it is the graph formed by arches that contain concepts that have these witnesses attached. The complete projection of the query in the first index can then be obtained from the inverted file. It is shown by the darkened part of the index of d_1 in figure 4.

8 RELIEF: An Image Retrieval System

Our image retrieval system is called RELIEF (a Relational Logical Approach based on Inverted Files) [21]. The system is implemented on top of the O_2 system [3]. The main component of the system is a conceptual graph platform allowing to perform all the graph operators. Our experimentations were done on an image test collection called PARYSIS [6], that was also used for test purposes with the systems developed in the framework of the European FERMI project [4]. The collection contains about 650 photographs taken on the beginning of the century and it is accompanied by 30 queries. The collection was originally indexed by specialists of the French Ministry of Culture, in the form of keywords attached with descriptive comments about the images. From these original indexes, manual indexing provided the corresponding indexes in the form of conceptual graphs, according to the image model introduced in [12]. The indexing process [6] used a concept lattice of about 7000 concepts and a relation lattice of 350 relations. The average indexing graph consists of 35 arches.

The users introduce their queries through the Web structured query interface presented in figure 8. This figure corresponds to the query: "*find the images that are street scenes and show an human in the front of a building*". The query is introduced using the three areas of the query formulation screen (see figure 8):

i) the query structural area (the upper region of the interface). In our example, the query searches for an image that is composed of two objects: "*human (humain)*" and "*facade*". As shown in figure 8, the user can consult $\mathcal{T}_c$.

ii) the query specification area (the middle region of the interface). It refines the query, in our case by specifying that the image type is "*street scene*". The corresponding arch is [IMAGE] $\rightarrow$ (Genre de l'image) $\rightarrow$ [SCENE-DE-RUE].

iii) the query relational area (the lower region of the interface). The symbolic and spatial (2D and 3D) relations are introduced here to provide more accuracy for image retrieval. Together with the second area, it allows for the extension of the query entered in the structural area. In our case, it contains the spatial relation "*before (devant)*" and the arch is [HUMAIN] $\rightarrow$ (devant-2D) $\rightarrow$ [FACADE].

When retrieval is started, the data introduced by the user are collected and an internal representation of the query is produced. The results are classified in relevance classes that correspond to the exact and partial projections respectively [21].

To evaluate our system, we considered another system, EMIR^2 [12], which also uses *CGs* as the underlying formalism and the classical projection operator as the matching function. The 30 FERMI queries were run on both RELIEF and EMIR^2. An overview of the results is shown in the figure 9. They prove the significant speed-up that we obtained.

For the analysis of the quality of answers, we also consider as reference the well-known SMART system [24, 25]. The input

[3] See the demonstration session of this conference.

[4] Esprit BRA 8134

data for SMART are the original keywords and textual descriptions of the images given by specialists [5]. The results of the comparison of our system to SMART and EMIR^2, in terms of precision/ recall, are shown in figure 10. They show that a simple formalism such as that used by SMART is not adequate to represent complex documents such as images. As compared to EMIR^2, our system is slightly superior, thanks to the treatment of the relations, as we mentioned in [21]. To evaluate the impact of relational treatment, we compared the results obtained with and without derivations on relations. Figure 11 shows its importance, and confirms that our extension to the *CGs* formalism improves significantly the quality of answers (about 10%).

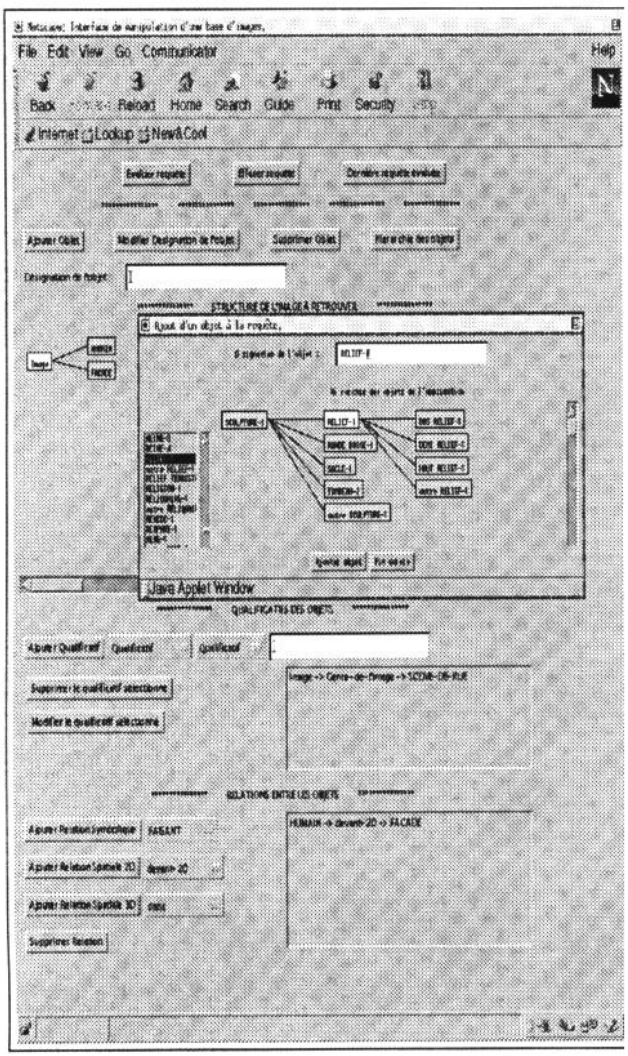

Figure 8: The French query formulation screen

Time (sec)	*EMIR²*	*RELIEF*
Min	6	1
Max	53	3
Average	18.8	1.8

Figure 9: Comparative execution times

The size of the disk space required by the objects modeling our structures is less than that of the objects modeling the

[5] We admit that the comparison with SMART is somewhat inappropriate, as the precision of the input index data is quite different. However, we use it in order to illustrate the limitations of keywords in terms of precision/ recall.

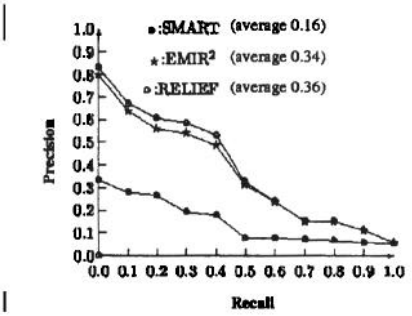

Figure 10: Comparative system precision/recall measures

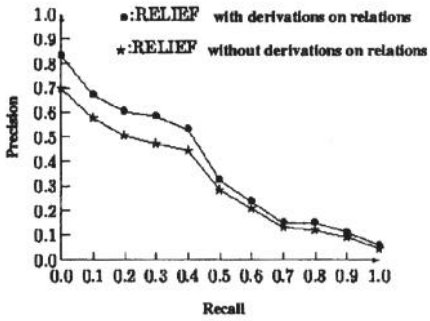

Figure 11: Importance of derivations on relations for the precision/recall measures

indexes. This is due to the implicit compression obtained using the inverted file, as only the first occurrence of the identical arches is represented, while the others determine the addition of witnesses only. Our inverted file takes about 3600 KB of disk space and the acceleration tables 4000 KB, while the objects modeling the conceptual graphs indexes occupy 10000 KB (see figure 12b)). We would like to mention that the indexes are no longer required once the structures are built, as the conceptual graphs can be re-generated from the inverted file. We obtain a reduction of the required space, which is indeed significant for a domain that deals with large amounts of data. However, for a larger test collection the space occupied would increase accordingly, and a solution has to be found to this issue.

Indexing has also shown another aspect, concerning the time required to build our structures. Though the impact on time is not as important for indexing as it is for retrieval, the results of figure 12a) suggest that an improvement is required. From the 500th index, the loading time increases exponentially. This is an implementation problem due to the fact that we did not use all possible optimizations in our first implementation. However, we are addressing this problem.

9 Conclusion

In this paper we proposed a solution to the challenge of using expressive and therefore expensive formalisms and keeping a reasonable retrieval time performance. Not only did we obtain fast retrieval, but we also improved the quality of answers, by extending the conceptual graphs formalism on a sound basis. Even though our solution is given for the conceptual graphs formalism, we believe that it is only an instance of a more general approach that can be applied to other expressive formalisms such as terminological logics [13], provided that adequate logical techniques are used. Our experiments were performed on a medium-sized image test collection. However, we consider the results to be suggestive, as the size of the indexes is im-

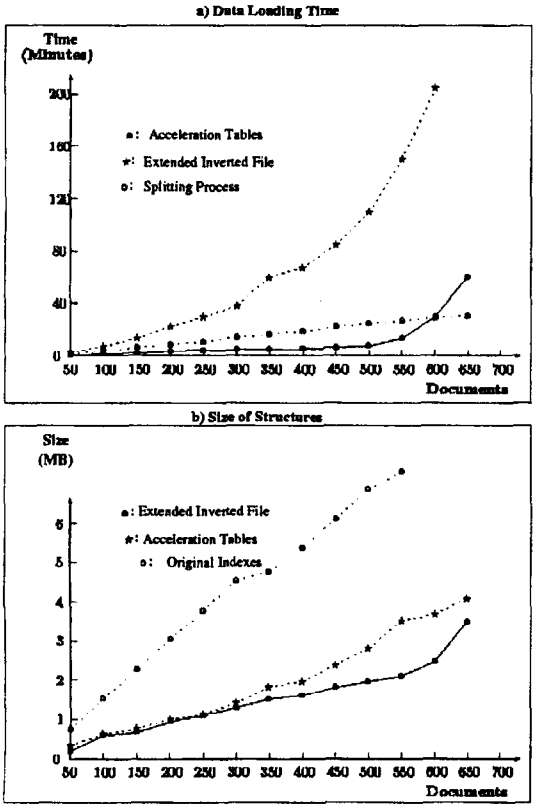

Figure 12: Time and disk space required to build our structures

portant. According to our estimations, a collection of 10000 images could be handled by the system with execution times no more than 20 seconds. This is very encouraging for a first implementation. Tests on larger collections is one of our immediate objectives. Presently, the construction of a large database of conceptual graphs in the *CGs* community is one of the purposes of the PEIRCE Workbench [6].

References

[1] G. Amati and I. Ounis. Interpreting conceptual graphs in first order logic: Projections, graph derivation systems and their complexity. Research report RAP98-001 (submitted), FUB (Rome), CLIPS-IMAG (Grenoble), 1998.

[2] E.W. Brown. Fast evaluation of structured queries for information retrieval. In *Proceedings the 18th Annual International ACM SIGIR Conference on Research and Development inInformation Retrieval, Seatle, USA*, pages 30–38, July 1995.

[3] C.C. Chang and H.J. Keisler. *Model Theory*. North-Holland, 1973.

[4] M. Chein and M.L. Mugnier. Conceptual graphs: fundamental notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[5] Y. Chiaramella and J.P. Chevallet. About retrieval models and logic. *The Computer Journal*, 35(3), 1992.

[6] Y. Chiaramella and M. Mechkour. Indexing an image test collection. Technical report, FERMI BRA 8134, April 1997.

[7] J. Farradane. Relational indexing Part I. *Journal of Information Science*, 1(5):267–276, 1980.

[8] 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.

[9] 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.

[10] H.J. Levesque and R.J. Brachman. *A fundamental tradeoff in knowledge representation and reasoning*, pages 42–70. Morgan Kaufmann Publishers, Los Atlos, California, 1985.

[11] H.J. Levesque and R.J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3(2):78–93, May 1987.

[12] 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.

[13] 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.

[14] M.L. Mugnier. *Contributions Algorithmiques pour les Graphes d'Héritage et les Graphes Conceptuels*. PhD thesis, Université Montpellier II, October 1993.

[15] M.L. Mugnier and M. Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'Intelligence Artificielle*, 10(1), 1996.

[16] J. Nie. An information retrieval model based on modal logic. *Information Processing & Management*, 25(5):477–491, 1989.

[17] J. Nie. *Un Modèle Logique général pour les systèmes de recherche d'informations. Application au prototype RIME*. Phd thesis, Université Joseph Fourier, Grenoble, 1990.

[18] I. Ounis. *Un modèle d'indexation relationnel pour les graphes conceptuels fondé sur une interprétation logique*. PhD thesis, Université Joseph Fourier, Grenoble, February 1998.

[19] 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.

[20] I. Ounis and M. Pasca. An extended inverted file approach for information retrieval. In *International Database Engineering and Application Symposium (IDEAS'97), Montreal, Canada*, pages 397–402. IEEE Computer Society Press, August 1997.

[21] 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.

[22] I. Ounis and M. Pasca. Effective and efficient relational query processing using conceptual graphs. In *20th Annual BCS-IRSG Colloquium on IR Research, Autrans, France*. EWIC, Springer-Verlag, 25–27 March 1998.

[23] P.F. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. McGregor, W.S. Mark, D.L. McGuiness, B. Nebel, A. Schmiedel, and J. Yen. Term subsumption languages in knowledge representation. *AI Magazine*, 11:16–23, 1990.

[24] G. Salton. *The SMART project*. Prentice Hall, 1971.

[25] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. Mcgraw Hill Book Company, New York, 1983.

[26] J.F. Sowa. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley Publishing Company, 1984.

[27] 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.

[28] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[6] The activities of the *CGs* community world-wide are presented in http://ksi.cpsc.ucalgary.ca/ lukose/cseic/