Query Evaluation and Information Fusion in a Retrieval System for Multimedia Documents

I. Glöckner and A. Knoll
Faculty of Technology, University of Bielefeld
P.O. 10 01 31, 33501 Bielefeld, Germany

Abstract *Despite their predominant application in robotics, the utility of methods for information fusion is not limited to sensor-based fusion tasks. The paper presents an information retrieval (IR) system for multimedia weather documents which makes use of linguistic fusion methods and a semantically rich retrieval model based on methods from fuzzy set theory. The computational problem of how to efficiently organize the query evaluation process is solved by object-based mediation and asynchronous parallel invocations both of the document evaluation and fusion methods.*

Keywords: Information fusion, information retrieval, mediators, multimedia systems

1 Introduction

Today's information search services do not fully exploit the wealth of information offered. One of the reasons is that the mutual contribution of a document's parts to its content are not considered, nor are relationships of documents (e.g. hyperlinks).

A number of attempts to utilize the rich document structure of hypertext documents for querying have been proposed, for example W3QL [1] and FLORID [2]. However, the structure (partitioning in sections etc.) of a document is only indirectly related to its content. In particular, users typically know the precise structure of the individual documents satisfying their information need only *after* having found these documents. Web query languages like WebSQL [3] support the search for hypertext links. But again, users querying the IR system know the precise (hyperlink) structure of desired documents only *after* they have found the relevant documents. Therefore a *gradual measure* of a pair of documents being related could prove useful, supported by methods for processing imprecise information.

Federated IR systems aim at providing uniform access to a number of networked and possibly heterogeneous information sources. A typical architecture for information integration is depicted in Fig. 1. It mediates access to a complex system of multiple and possibly very heterogeneous information sources through "wrappers" in such a way that the illusion of a local database with rich informational content emerges. The results of the individual wrappers are merged by the *mediator* component [4] into a global logical view. Examples of such systems are HERMES [5], SIMS [6], and TSIMMIS [7].

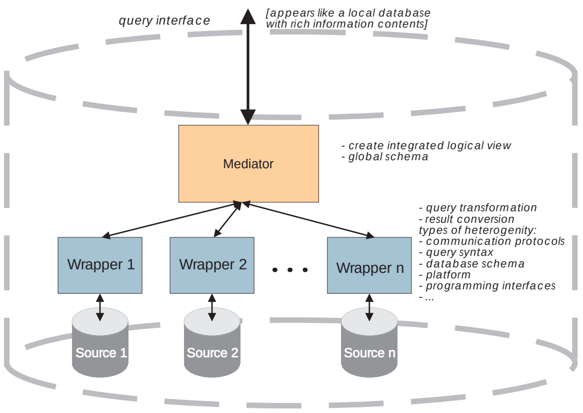

Figure 1: Information integration architecture

In the evolving field of *content-based image retrieval* [8, 9, 10], images are analysed for features like structure, color distribution (histograms or correlograms), texture etc., which all correspond to the signal level of the image (as opposed to the semantic level). Like string matching in text retrieval, these methods are not restricted to specific domains. However, the results obtained are not yet comparable to that of text retrieval.

The low filter quality of today's generic techniques for multimedia retrieval suggests another strategy for building high-quality search services for multimedia documents, namely that of combining a substrate of generic methods for document description and information fusion with domain-specific methods which are taylored to a chosen field of application. The current concept of broad-coverage search engines is thus contrasted with that of a search service specialized to a topic area of general interest, such as weather, geography, sports, or vacation, which in this area provides search facilities on a new level of quality. These considerations lead to the following profile of a *high performance query server* (HPQS):

- *natural language (NL) interface*, to help occasional users formulate their search interest;
- *on-line search* of the document base under the user query: the complex modes of NL querying may frequently not be anticipated through pre-computed descriptors and necessitate the application of direct-search methods;
- *scalability:* acceptable response times must be ensured even for large data sets;
- *evaluation and combination* of pieces of information extracted from different sources, by applying methods for information fusion.

In the following section, we shall briefly introduce the HPQS system, and then concentrate on aspects of information fusion and query mediation.

2 The HPQS system

Fig. 2 depicts the architecture of the HPQS system [11]. The user interacts with the system via a graphical user interface (Java applet); natural language queries are typed into a query mask using the keyboard.[1] The morphological and syntactical analyses are carried out by the *natural language interface*, which generates a semantical representation of the query content. This representation is purely declarative, i.e. not directly executable. The subsequent *retrieval module* hence applies domain specific transformation rules which translate the declarative representation into a sequence of executable database queries. These trigger the generic evaluation and information fusion functionality as well as additional application methods. Execution of the generated database queries is controlled by the *multimedia mediator* which optimizes response times by maintaining a cache for storage and reuse of intermediate search results. The use of a *parallel media server* coupled with *dedicated high-speed VLSI processors* for image and text search ensures acceptable response times even when a computationally expensive online analysis of the mass data has to be performed.

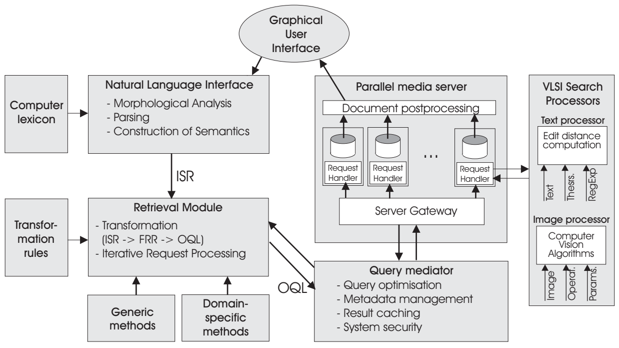

Figure 2: Architecture of the HPQS system

As the prototypical application of HPQS, we have chosen meteorological (weather information) documents. The range of meteorological documents used in our system comprises textual weather reports (ASCII and HTML), as well as satellite images and various weather maps (colour images). Query types in this application scenario include the following:

- What is the weather like in Bielefeld?
- Is it more often rainy on Crete than in southern Italy?
- Show me pictures of cloud formation over Bavaria!
- In which federal states of Germany has it been humid but warm last week?
- There were how many sunny days in Berlin last month?

The system accepts questions in exactly this form as text strings.

[1] i.e. speech input is not yet supported.

3 Formal retrieval representation

The retrieval component of the HPQS system utilizes a formal retrieval representation (FRR) which combines generic FRR methods (search techniques for documents of all relevant media and methods for information fusion) and domain-specific methods (which implement domain concepts). The FRR is syntactically identical to ODMG-OQL (Object Query Language); the FRR functionality is provided by generic and application-specific classes in the object-oriented database schema of the mediator. The generic part of FRR comprises:

- an elaborate *text-search component* (based on the dedicated VLSI processors for approximate full-text search);
- *image analysis primitives* (partly implemented in VLSI hardware);
- *discrete and parametrized fuzzy sets* and corresponding connectives from fuzzy set theory;
- *fuzzy quantifiers* which provide a numerical interpretation of quantifying expressions in NL queries.

Fuzzy quantifiers also prove useful in weighted information fusion tasks, i.e. for combining pieces of information according to numerical degrees of relevance (see below).

The generic FRR can be extended by domain-specific methods, which provide an interpretation for NL domain concepts based on the raw document data. The HPQS prototype has been taylored to the meteorology domain by implementing cartographic projections of the considered image classes; objective ("more than 20 degrees") and subjective ("warm") classification of temperature readings; estimation of cloud-top height and cloud density in satellite images; determination of degrees of cloudiness ("sunny"); and other domain concepts. In the same way that text-matching provides only a very coarse, but often still useful, approximation of text-understanding, we attempt to model only that portion of the domain concepts which must be captured to restrict the search to useful query results.

Table 1 displays the FRR sequence generated for an example query. The results of the query are shown in Fig. 3.[2]

Generated FRR

```
q_311:  element(select x.shape from x in FederalStates
where x.name = "Bavaria")
q_312:  select i from i in MeteoFranceImages where
i.date.ge(1997,8,1,0,0,0) and i.date.lower(1997,8,8,0,0,0)
q_313:  select i.pred from i in q_312 where i.pred <> i
q_314:  select ImageAndRelevance(image:i,
relevance:q_311.rateGreaterEqual(0.7, i.cloudiness().
sunny().negation().germanyProjection())) from i in q_312
q_315:  select ImageAndRelevance(image:i,
relevance:q_311.rateGreaterEqual(0.7,
i.cloudiness().sunny().germanyProjection()))
from i in q_313
q_316:  select ImageAndRelevance(image:i.image,
relevance:i.relevance.min(j.relevance))
from i in q_314, j in q_315
where j.image = ((HpqsMeteoFranceImage)i.image).pred
q_317:  select f.relevance from f in q_316
q_318:  select f from f in in q_317 order by 1
q_319:  HpqsGreyValSeq(greyval_sequence:
o2_list_GreyVal(q_318)).determineThreshold()
q_320:  select ImagesAndRelevance(image:f.image,
pred:((HpqsMeteoFranceImage)f.image).pred,
succ:((HpqsMeteoFranceImage)f.image).succ,
relevance:f.relevance)
from f in q_316 where f.relevance.ge(q_319) = 1
```

Table 1: FRR sequence generated for query: *"Show me pictures of cloud formation over Bavaria in the first week of August 1997!"*

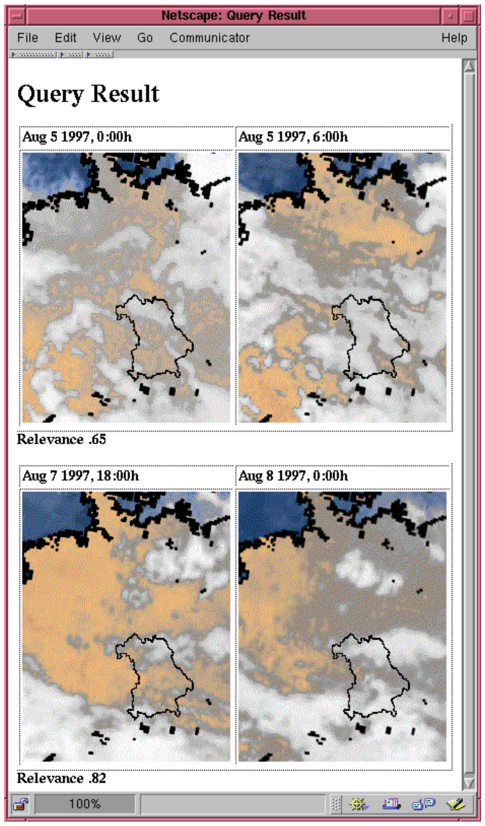

Figure 3: Result of example query

[2]see [11] for a description of the search process.

4 Linguistic information fusion

In [12], we have pointed out that providing natural language access to a multimedia retrieval system cannot be accomplished merely by adding an NL frontend to an existing retrieval "core". This is because the modes of information combination expressible in natural language are not restricted to the Boolean connectives supported by traditional retrieval systems. In particular, vague quantifying expressions (fuzzy quantifiers) like *most*, *almost everywhere*, are often used in NL queries to express accumulative criteria such as *"almost all of Southern Germany is cloudy"*. In this example, we have a set E of pixel coordinates. Each pixel $e \in E$ has an associated relevance $\mu_{X_1}(e) \in \mathbf{I} = [0, 1]$ with respect to the fusion task, which in this case expresses the degree to which pixel $e \in E$ belongs to Southern Germany, and each pixel has an associated evaluation $\mu_{X_2}(e) \in \mathbf{I}$ which expresses the degree to which the pixel is classified as cloudy (see Figs. 4 and 5). The mappings $\mu_{X_1}, \mu_{X_2} : E \longrightarrow \mathbf{I}$ can be viewed as membership functions representing fuzzy subsets $X_1, X_2 \in \widetilde{\mathcal{P}}(E)$ of E, where $\widetilde{\mathcal{P}}(E)$ is the fuzzy powerset of E. Our goal is to provide a mapping $\widetilde{Q} : \widetilde{\mathcal{P}}(E) \times \widetilde{\mathcal{P}}(E) \longrightarrow \mathbf{I}$ which, for each considered satellite image, combines these data to a numerical result $\widetilde{Q}(X_1, X_2) \in \mathbf{I}$ as requested by the NL expression "almost all".

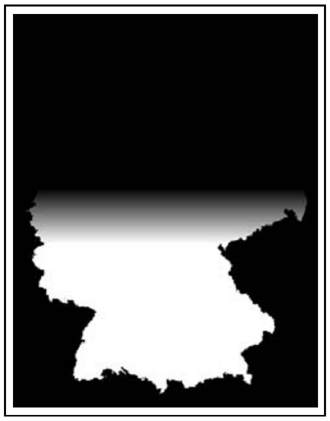
Figure 4: A possible definition of X_1 = southern_germany (Pixels with $\mu_{X_1}(e) = 1$ are depicted white)

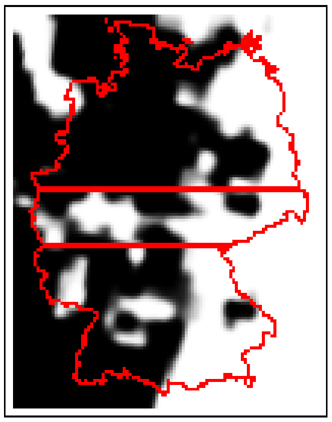
Figure 5: Fuzzy image region X_2 = cloudy (Pixels classified as cloudy are depicted white. The contours of Germany, split in southern, intermediate and northern part, have been added to facilitate interpretation)

Apparently, an operator which implements "almost all" yields adequate results only if it captures the *meaning* of "almost all". We have therefore decided to base our solution to the fusion problem on (a) the Theory of Generalized Quantifiers (TGQ [13]), which has developed important linguistic concepts for describing the meaning of NL quantifiers; and (b), methods from fuzzy set theory, known as *fuzzy linguistic quantifiers* [14, 15], which are concerned with aspects of fuzziness involved, i.e. the use of concepts without sharply defined boundaries ("Southern Germany", "cloudy", "almost all"). Our investigation of existing approaches to fuzzy quantification [14, 16, 17] based on criteria of TGQ has led us to reject these approaches because of their inconsistency with linguistic facts. Building on TGQ, we have formulated a set of axioms which characterizes mathematically sound models of fuzzy quantification; in addition, we have presented a model of the axioms [18]. In [19], we have shown that this approach is computational by presenting a histogram-based algorithm for the efficient evaluation of the resulting operators.

In our system, we are currently using these operators for the fusion of fuzzy sets of pixels (local quantification) or fuzzy sets of time points (temporal quantification), see Table 2. We are hence utilizing spatio-temporal relationships between extracted pieces of information in order to compute a combined evaluation of the documents

Quantification over local regions
few clouds over Italy
many clouds over southern Germany
more clouds over Spain than over Greece
cloudy in Northrine-Westphalia (implicit)
Quantification over regions in time
almost always cold in the last weeks
more often sunny in Portugal than in Greece
hot in Berlin in the previous week (implicit)

Table 2: Examples of fuzzy quantification in the meteorology domain

of interest. This type of relationship might look different from those established by hypertext links, and from intra-document relationships (between parts of a composite document). However, all of these relationships can be deployed for retrieval purposes only if suitable methods for information fusion are available. Fuzzy quantifiers are promising in this respect because they are both human-understandable and sufficiently powerful to handle the required two-dimensional fusion problem (data to be combined plus weights of relevance). The basic aptitude of fuzzy quantifiers for combining search ratings of a document's parts to a global evaluation has recently been demonstrated by Bordogna&Pasi [20].

5 Mediation and query evaluation

In the HPQS system, we have only one (but a very complex) information source, viz. the parallel media server. The tasks of the HPQS mediator include:

- abstraction from details of the parallel media server, e.g. socket-based communication protocol and query syntax;
- making optimal use of the *parallelism* available in the external source;
- establishing a well-structured view of the multimedia system, which to the retrieval module (the mediator's client) should appear like an object-oriented ODMG database;
- maintenance of a *proxy state* of the external document base: method invocations can only be delegated to the parallel media server if the documents to which these methods should be applied are known to the mediator;
- *materialization* of results of method invocations, in order to avoid redundant computations by reusing query results of a result cache.

The efficient organisation of method invocations on the external source is of particular importance to the HPQS system because a large number of documents (and hence of instances of document evaluation and information fusion tasks) must be processed with acceptable response times. The problem is that the database executes OQL queries sequentially, and cannot directly benefit from the parallel processing abilities of the media server.

The first HPQS mediator, described in [21], makes use of blockwise request execution in order to benefit from the parallelism in the media server source. The transformation of ODMG-OQL queries to the mediator into simpler queries which can be executed in parallel will be illustrated by an example. The mediator might e.g. receive the query

```
select ImageAndRelevance( image : I,
   relevance :
      BAY.rateGreaterEqual(0.7,
      I.cloudiness().sunny()
      .negation()
      .germanyProjection()))
   from I in q_18
```

By means of query transformations, it decomposes the query in a sequence of elementary queries:

```
R1: select I.cloudiness()
    from I in q_18
R2: select I.sunny() from I in R1
...
```

These simple queries are transformed into blocks of requests and transmitted to the media server, which executes them in parallel and returns the set of results to the mediator. Using such blockwise parallel calls, the example query is executed as depicted in Figs. 6 and 7. The nodes (circles) represent elementary requests (individual method invocation given on particular choice of parameters). The dependency structure of the requests is represented by arcs (a complex expression depends on its subexpressions in the sense that it can only be evaluated once each of its subexpressions have been evaluated).[3]

In the figures, we have assumed that nine images are to be processed and that eight processing nodes are available on the parallel server. With blockwise evaluation, execution starts which the block request to compute $x.\texttt{cloudiness}()$ for the nine images (requests $A_1 \dots A_9$), which is sent to the parallel server; the mediator then suspends processing until the parallel server returns the results for the whole block of requests. Having obtained the results of the first block, the mediator then initiates processing of the second block $B_1 \dots, B_9$, to compute $y.\texttt{sunny}()$ on all results y of the first block, etc. As witnessed by Figs. 6 and 7, this

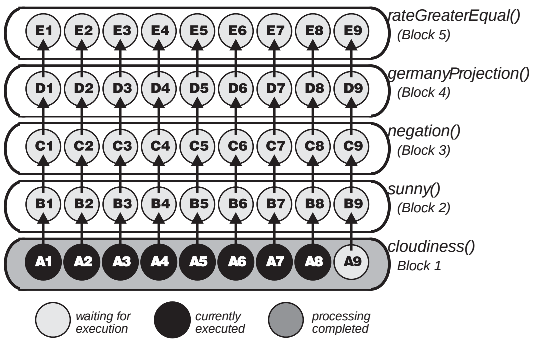
E1 E2 E3 E4 E5 E6 E7 E8 E9
rateGreaterEqual() (Block 5)
D1 D2 D3 D4 D5 D6 D7 D8 D9
germanyProjection() (Block 4)
C1 C2 C3 C4 C5 C6 C7 C8 C9
negation() (Block 3)
B1 B2 B3 B4 B5 B6 B7 B8 B9
sunny() (Block 2)
A1 A2 A3 A4 A5 A6 A7 A8 A9
cloudiness() Block 1
waiting for execution
currently executed
processing completed

Figure 6: Blockwise Parallel Execution A

blockwise parallel evaluation does not make optimal use of the computing resources. Assuming that each request in the first block needs about 10s processing time, the parallel server will execute the requests $A_1 \dots A_8$ in 10s. However, it needs another 10s to process request A_9 (Fig. 7). Only after 20s, the result of the block can be sent to the mediator, and processing of the second block can be initiated. This behaviour is suboptimal because when executing A_9, only one work node is active, and the other seven work nodes are idle, although the results of $A_1 \dots A_8$ are available so that execution of $B_1 \dots B_8$ could be started.

[3]In our example, the dependency structure is a chain, but with multiplace functions, it becomes a forest (set of trees). If intermediate results are re-used by a caching mechanism (as is done by the mediator), the structure becomes a directed acyclic graph.

The blockwise evaluation approach requires the mediator to parse OQL queries and reformulate these into blocks of requests to the media server.

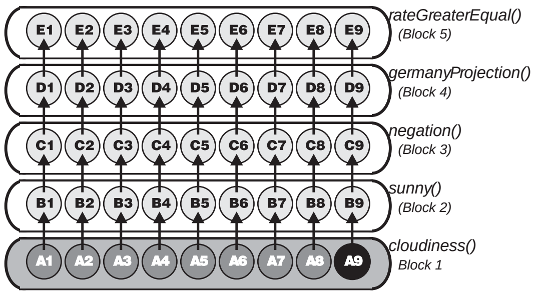
E1 E2 E3 E4 E5 E6 E7 E8 E9
rateGreaterEqual() (Block 5)
D1 D2 D3 D4 D5 D6 D7 D8 D9
germanyProjection() (Block 4)
C1 C2 C3 C4 C5 C6 C7 C8 C9
negation() (Block 3)
B1 B2 B3 B4 B5 B6 B7 B8 B9
sunny() (Block 2)
A1 A2 A3 A4 A5 A6 A7 A8 A9
cloudiness() Block 1

Figure 7: Blockwise Parallel Execution B

In order to avoid the intricacies of OQL analysis and translation, and to make better (i.e. more fine-grained) use of the parallel computing resources, we have decided to build an alternative mediator for the HPQS system based on parallel asynchronous method invocations. This approach rests on the following considerations. We can leave the database application unchanged (i.e. still executing sequentially) and still profit from parallel execution on the media server only if the act of initiating or triggering a request is *decoupled* from the processing of the request. In the alternative medi-

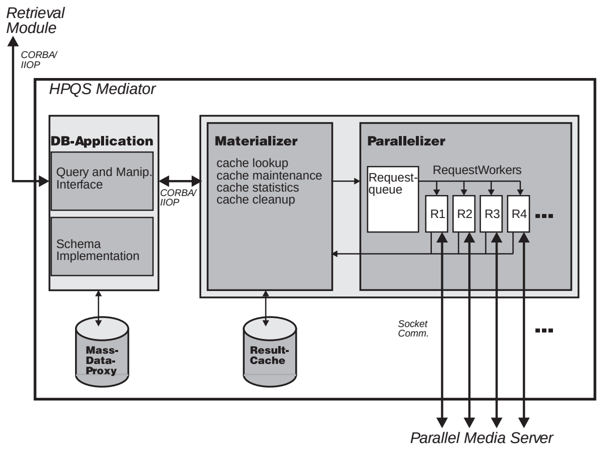

Figure 8: Asynchronous Execution Architecture

ator (see Fig. 8), we have the database *trigger* the requests sequentially: triggering is a non-blocking call which immediately returns with a *result key*. If the request cannot be found by the materializer in its result cache, it is inserted into a request queue. The parallelizer makes use of a number of `Re-`

questWorkers (one for each processor node of the parallel server) which fetch requests from the queue and cater for their execution on the parallel server.

It is sufficient for the database to know the result key to initiate further requests. Only when direct access to the computed result is necessary (e.g. in order to display a result image), it performs a "fetch" call on the result key to obtain the computed data. These fetch calls are blocking and wait until the result is available.

Snapshots of the parallel asynchronous execution of the example query are presented in Figs. 9, and 10. The database executes the query (i.e. triggers requests) using its "normal" execution order, which respects the dependencies of the requests. The requests are hence triggered, and inserted into the request queue, as indicated by the small numbers beneath the nodes in Fig. 9. The policy for obtaining the request to be executed from the queue is to select the "oldest" request all dependencies of which are satisfied (in the sense that the results for all arguments are available). The precise execution order with the parallel asynchronous evaluation strategy is hence dependent both on the insertion order into the queue and on the termination order of requests as processed by the parallel server. The initial configuration of processed requests as depicted in Fig. 9 is typical because the database triggers much faster than the requests can be executed. A later processing state is depicted in Fig. 10.

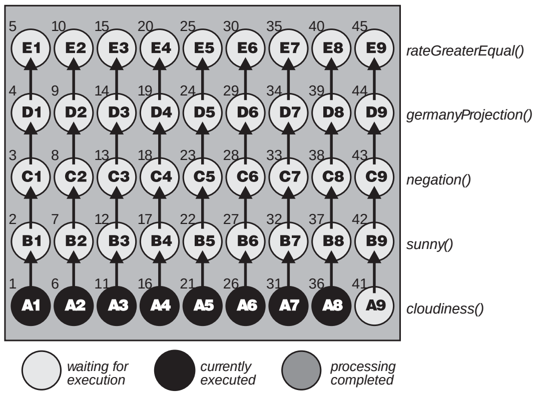

Figure 9: Asynchronous Parallel Execution A

When the processing of a request is completed, the corresponding RequestWorker immediately selects the next request to be processed from the queue. We achieve a better utilisation of the parallel computing resources because it it avoided that processor nodes be idle.

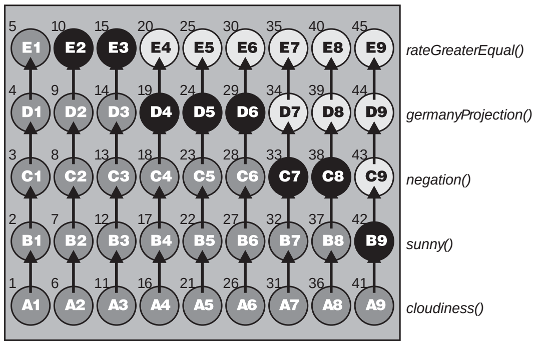

Figure 10: Asynchronous Parallel Execution B

6 Discussion

We have presented a system architecture suitable for building high-quality multimedia search services for restricted (but in principle arbitrary) topic areas. By providing an NL interface, technical barriers in accessing the system are removed. The imprecision and vagueness of NL queries must be handled because an adequate system behaviour can only be achieved if these factors do not result in system failure or implausible results. We have therefore developed a semantically rich retrieval model based on methods methods from fuzzy set theory. Emphasis has been put on linguistic methods for information fusion (viz. fuzzy quantifiers). Apart from our use of these methods to utilize spatio-temporal relationships, such methods are a prerequisite of combining the contents spread over the parts of a multimedia document, and of utilizing relationships established by hypertext links in a broad range of other applications.

HPQS supports online search and thus offers versatile ways of querying: there is no restriction to pre-computed descriptors and their Boolean combinations. We have combined several techniques in order to ensure acceptable response times, in particular *parallelisation of method invocations*, by utilizing a parallel asynchronous evaluation strategy, and the use of *materialization*, which yields a speed-up for frequent queries (or subqueries) comparible to that of traditional indexing.

Although HPQS makes use of only one data source (the parallel media server), the information provided by the various document types is partially overlapping. For example, satellite images of different weather satellites (Meteosat, NOAA) or weather maps of different meteorological services can all be used to compute estimates of the degree of cloudiness at a given geographical location, and the results obtained can either support each others or contradict. Existing mediators like HERMES [5] have chosen to handle such cases by conflict resolution rules which specify a priority ordering on the sources, in order to *select* one of the conflicting pieces of information. We are currently working on the problem of *combining* (rather than selecting) such overlapping and possibly contradictory data, based on our linguistic methods of information fusion.

References

[1] http://www.cs.technion.ac.il/~konop/w3qs.html.

[2] B. Ludäscher, R. Himmeröder, G. Lausen, W. May, and C. Schlepphorst. Managing semistructured data with FLORID. *Information Systems*, 23(8):589–612, 1998.

[3] http://www.cs.toronto.edu/~websql/.

[4] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[5] http://www.cs.umd.edu/projects/hermes/.

[6] http://www.isi.edu/sims/.

[7] http://www-db.stanford.edu/tsimmis/tsimmis.html.

[8] A. Del Bimbo and P. Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 19(2):121–132, 1997.

[9] V. Castelli, L. Bergman, C. Li, and J. Smith. Search and progressive information retrieval from distributed image/video databases: the SPIRE project. In Nikolaou and Stephanidis [22].

[10] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkhani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9), September 1995.

[11] A. Knoll, C. Altenschmidt, J. Biskup, H.-M. Blüthgen, I. Glöckner, S. Hartrumpf, H. Helbig, C. Henning, Y. Karabulut, R. Lüling, B. Monien, T. Noll, and N. Sensen. An integrated approach to semantic evaluation and content-based retrieval of multimedia documents. In Nikolaou and Stephanidis [22], pages 409–428.

[12] I. Glöckner and A. Knoll. Fuzzy quantifiers for processing natural-language queries in content-based multimedia retrieval systems. TR97-05, Technische Fakultät, Universität Bielefeld, 1997.

[13] J. Barwise and R. Cooper. Generalized quantifiers and natural language. *Ling. and Phil.*, 4:159–219, 1981.

[14] L.A. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Computers and Math. with Appl.*, 9:149–184, 1983.

[15] Y. Liu and E.E. Kerre. An overview of fuzzy quantifiers. (I). interpretations. *Fuzzy Sets and Sys.*, 95:1–21, 1998.

[16] A.L. Ralescu. A note on rule representation in expert systems. *Information Sciences*, 38:193–203, 1986.

[17] R.R. Yager. Families of OWA operators. *Fuzzy Sets and Systems*, 59:125–148, 1993.

[18] I. Glöckner. DFS – an axiomatic approach to fuzzy quantification. TR97-06, Techn. Fakultät, Univ. Bielefeld, 1997.

[19] I. Glöckner, A. Knoll, and A. Wolfram. Data fusion based on fuzzy quantifiers. In *Proc. of EuroFusion98*, pages 39–46, 1998.

[20] G. Bordogna and G. Pasi. A fuzzy information retrieval system handling users' preferences on document sections. In D. Dubois, H. Prade, and R.R. Yager, editors, *Fuzzy Information Engineering*. Wiley, 1997.

[21] J. Biskup, J. Freitag, Y. Karabulut, and B. Sprick. A mediator for multimedia systems. In *Proceedings 3rd International Workshop on Multimedia Information Systems*, Como, Italia, Sept. 1997.

[22] C. Nikolaou and C. Stephanidis, editors. *Research and Advanced Technology for Digital Libraries: Proceedings of ECDL '98*, LNCS 1513. Springer, 1998.