Journal of Universal Computer Science, vol. 2, no. 7 (1996), 515-532
submitted: 5/2/96, accepted: 23/2/96, appeared: 28/7/96 © Springer Pub. Co.

On Combining the Knowledge of Heterogeneous Information Repositories

Uwe M. Borghoff
Rank Xerox Research Centre, Grenoble Laboratory
6, chemin de Maupertuis. F-38240 Meylan, France.
Email: borghoff@grenoble.rxrc.xerox.com

Johann H. Schlichter
Institut für Informatik, Technische Universität München
D-80290 München, Germany.
Email: schlicht@informatik.tu-muenchen.de

Abstract: The Internet facilitates access to a large amount of electronic information. However, in order to exploit the flood of information, sophisticated search facilities are needed which convert the inundation of electronic data coming from numerous sources into real knowledge. From this knowledge a whole range of users will benefit, from business people to casual surfers and shoppers on the Internet.
Intelligent agents or knowledge brokers play a vital role in realizing this vision. This paper presents a framework for knowledge brokers who search for information which is potentially available but stored in a way not always foreseen how the information will be exploited. More striking, the paper presents an architectural framework where the user can retrieve and combine knowledge uniformly, irrespective of where or how the knowledge-representing information is stored.
Lessons learned from a prototype implementation allow a discussion of shortcomings due to the emphasis of current information repositories and their interfaces, above all their poor support for knowledge combination and the difficulty of localizing the appropriate information repositories.

Key Words: distributed agent system, knowledge brokers, information gathering, data retrieval, Internet.
Category: H.2.5., H.3.

1 Introduction

Since their origins, database systems have provided data storage and data retrieval in a uniform way. These systems have emphasized data consistency and data coherence. Data have been stored in a way which has allowed appropriate searches with well-defined formatted retrieval languages. Roughly speaking, the same still holds to be true within the context of federated databases [Sheth and Larson, 1990]. Federated databases hide their different data models and query languages. Through schema integration and a common query language (e. g. SQL), the federation appears as a logically integrated database of existing heterogeneous distributed databases.

New electronic information sources, such as distributed file systems, Internet-based information sources, or other on-line information repositories proliferate. These distributed "collections" of information have a slightly different orientation. They are used to present data, sometimes are tailored to present the information graphically, rather than storing them for retrieval. In the Internet many people now include their private collections of data for public use. Large data collections of bibliographical information, e. g. in BibTeX format, are accessible electronically. In general, a variety of miscellaneous information is privately updated, gathered for special needs, and provided to the large user community. In fact, if a user is looking for a particular piece of information, there is a good chance that this information is stored somewhere in publishing tools like Gopher or the World-Wide Web. However, it is difficult to discover the location and the way in which the information is provided. Since search forms or other discovery tools are developed independently, interoperability is difficult to achieve. In many cases, moreover, a user must contact more than one source to reach the final destination where the answer to his/her question can be found.

To combine search results, to extract knowledge from one search in order to trigger a subsequent search, and, finally, to evaluate the search hits, are major challenges in searching heterogeneous information repositories of the Internet. This paper presents an architectural framework to tackle these challenges.

The paper is organized as follows. After a brief discussion of related work in Sect. 2, we present in Sect. 3 the architectural framework which we used for knowledge combination. The prototype architecture consists of a user interface, knowledge brokers which specify electronic information through constraints, and wrappers which provide the interfaces to existing external data sources. Sect. 4 discusses shortcomings of the current approach. Sect. 5 concludes the paper.

2 Related Work

Well-established publishing systems, like Gopher and the World-Wide Web, provide a seamless information space in the Internet, at least as far as graphical browsing is concerned. Index and search subsystems appeared hand in hand with the rapid growth in the amount of information and in the number of users having specific needs. [Obraczka et al., 1993] and [Schwartz et al., 1992] give an overview of resource discovery approaches.

Some of the earliest Internet indexing approaches were the Wide-Area Information Servers (WAIS) [Kahle and Medlar, 1991], providing a Z39.50-based search and retrieval interface, and Archie [Emtage and Deutsch, 1992]. Archie periodically contacts a set of registered servers to gather a file index. Similar to that is Aliweb [Aliweb, URL] which contains user-written summaries of server contents that are displayed on request.

More recently, with Glimpse (GLobal IMPlicit SEarch) [Manber and Wu, 1994; Glimpse, URL] an index/search subsystem has been installed that allows sophisticated searches over entire file systems. Among other things, it allows misspelling and regular expression searches over non-uniform information including many types of documents. At the University of Karlsruhe, a prominent application has been realized on top of Glimpse, namely the sophisticated search facility for a large collection of computer science bibliographies [CS_biblio, URL].

Although multi-source index/search subsystems have already been built for Gopher, with Veronica [Veronica, URL], and for the Web, with Alta Vista [Alta Vista, URL], Lycos [Mauldin and Leavitt, 1994; Lycos, URL], and the World-Wide Web Worm [McBryan, 1994; WWWW, URL], retrieval engines or retrieval support systems for heterogeneous information are still open research fields [Barbara, 1993]. Early prototypes have however gotten an airing. The system Inquery

[Callan et al., 1992; Callan et al., 1995], currently being developed at Amherst University, calculates the appropriateness of heterogeneous information sources with respect to a given query. It chooses the best fitting sources and conducts the search processes. At Stanford University gGLOSS (generalized Glossary-Of-Servers Server) [Gravano and Garcia-Molina, 1995] addresses a similar idea. It keeps sophisticated statistics on available databases to determine an estimate of which databases are most appropriate for a given query. The search process is performed through a ranked list of databases. In contrast to Archie, which gathers an index without having a particular query in mind, Inquery and gGLOSS provide their indexes dynamically and are tailored to individual needs, via a single query. The indexes then guide individual searches across the set of servers.

As soon as appropriate index/search prototypes were implemented, intelligent agents [CACM, 1994; Wooldridge and Jennings, 1995] or knowledge brokers [Barbara and Clifton, 1992] started to exploit these subsystems. Harvest [Bowman et al., 1994a; Harvest, URL], for example, exploits as an index/search subsystem, both Glimpse and Nebula [Bowman et al., 1994b]. Knowledge brokers are autonomous entities that may collaborate, negotiate, and coordinate, but which by no means can be coerced into activities such as searching information or answering a query whose scope does not conform to the broker's ability in query handing (for more details see [Andreoli et al., 1995]). Thus, knowledge brokers are generally used in combination with index/search subsystems.

In the Constraint-Based Knowledge Broker model (CBKB) constraints have been introduced to flexibly manage the search space of broker agents as well as to flexibly adapt user requests and answers from information providers. [Andreoli et al., 1996] presents the theoretical background of CBKB. Protocol issues within CBKB are addressed in [Arcelli et al., 1995; Borghoff et al., 1996b]. [Fikes et al., 1995] also use logic-based models to capture the domain of expertise of information brokers. Rather than using constraints, their modeling language is based on a predicate logic with contexts. The Tsimmis project [Chawathe et al., 1994] takes a different approach using a self-describing object model for the internal representation of information and queries.

3 Knowledge Combination

We have developed a framework for knowledge brokers which searches information in a wide variety of heterogeneous information repositories and which combines the search results into user-customizable knowledge. During individual searches a variety of index/search subsystems are contacted, a variety of protocols are used, and different information formats are homogenized. The implemented prototype system is based on a distributed architecture following the client/server model.

The developed framework differs from other approaches for agent-based information gathering on the Web not only in the technology, but also in some fundamental assumptions. We differ from those approaches which view the Web as a kind of global market where agents – roaming over open electronic domains – will meet and gather information, possibly leading to business transactions. On the contrary, we see the Web as evolving into a galaxy of intranets, linking together information providers and users around common interests. On the basis of these social and economic considerations, technological choices can be consequently specialized to optimally fit the requirements of specific intranets and user communities. One such case of specialization of our framework, namely the adaptation of an agent infrastructure for constraint-based information gathering to the requirements of a network publication system for research and education, has been presented in [Borghoff et al., 1996c].

3.1 Architecture

As emphasized in Fig. 1, the architecture consists of four major components: user interface, knowledge brokers, wrappers and external databases. Note that the notion of database is used in a loose sense. It should be read in a comprehensive meaning, containing "real" database management systems but also distributed file systems, Internet-based information sources, or other on-line information repositories. The important part is the existence of some sort of index/search subsystem.

For the discussion of the architecture and its features only the first three components (user interface, knowledge brokers and wrappers) are of interest. With respect to the backends (external databases) we rely on already existing databases, e. g. databases accessible through the World-Wide Web, or locally available databases such as the Xerox phonebook. The external databases which are integrated into the prototype system cover a wide range of different information types ranging from collections of bibliographies and books to databases of telephone numbers for Xerox employees to listings of opera performances. In this heterogeneous environment some databases incorporate highly structured information while the information of other databases is completely unstructured.

The goal of the prototype system was the integration of heterogeneous databases storing different types of information, using different data representations, supporting different query languages and different interface protocols, into one environment. Among the different interface options and corresponding access methods, we especially focus on SQL databases, WAIS, and FTP, Gopher or HTTP servers in the case of databases accessible through the World-Wide Web. The users retrieve and combine knowledge uniformly, no matter where and how the knowledge-representing information is stored. The main emphasis was put on constraint-based knowledge combination. Thus, the approach using federated databases [Sheth and Larson, 1990] is not sufficient for our goal. Similarly the Teamwork approach proposed by Denzinger [Denzinger, 1995] applies knowledge combination to distributed search problems whose descriptions offer no natural way of dividing them a priori into subproblems.

3.2 Implementation

The graphical user interface has been implemented as Java applet which make the system queryable from standard Web browsers with applet support. The format of the queries/subqueries sent to the brokers is based on constraints over feature structures. Adapters are provided to convert this format to human readable graphical widgets on the users side.

The core of the system is given by the brokers, which provide various important services such as intelligent cacheing and knowledge combination. The current implementation uses ForumTalk. Brokers are implemented as ForumTalk

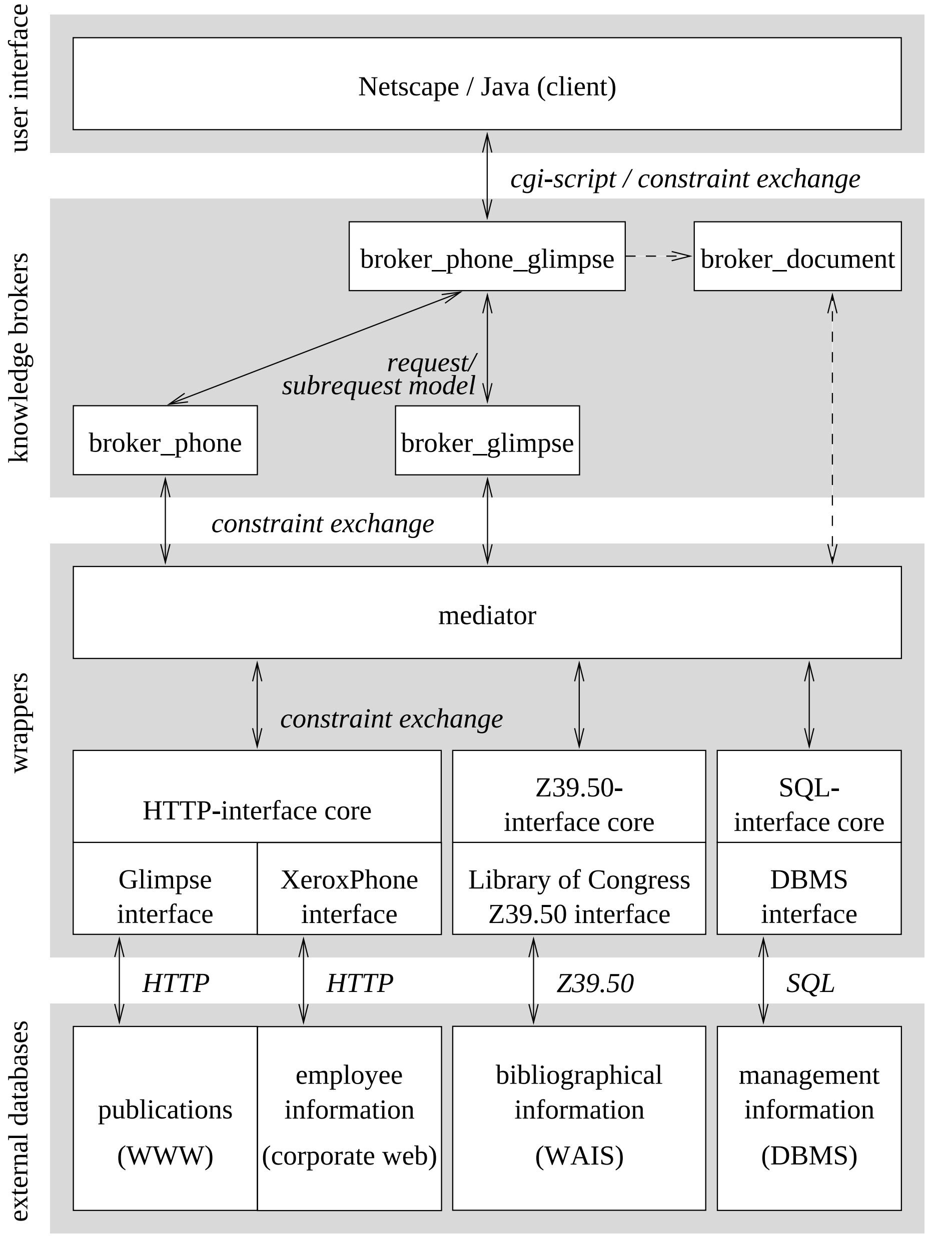
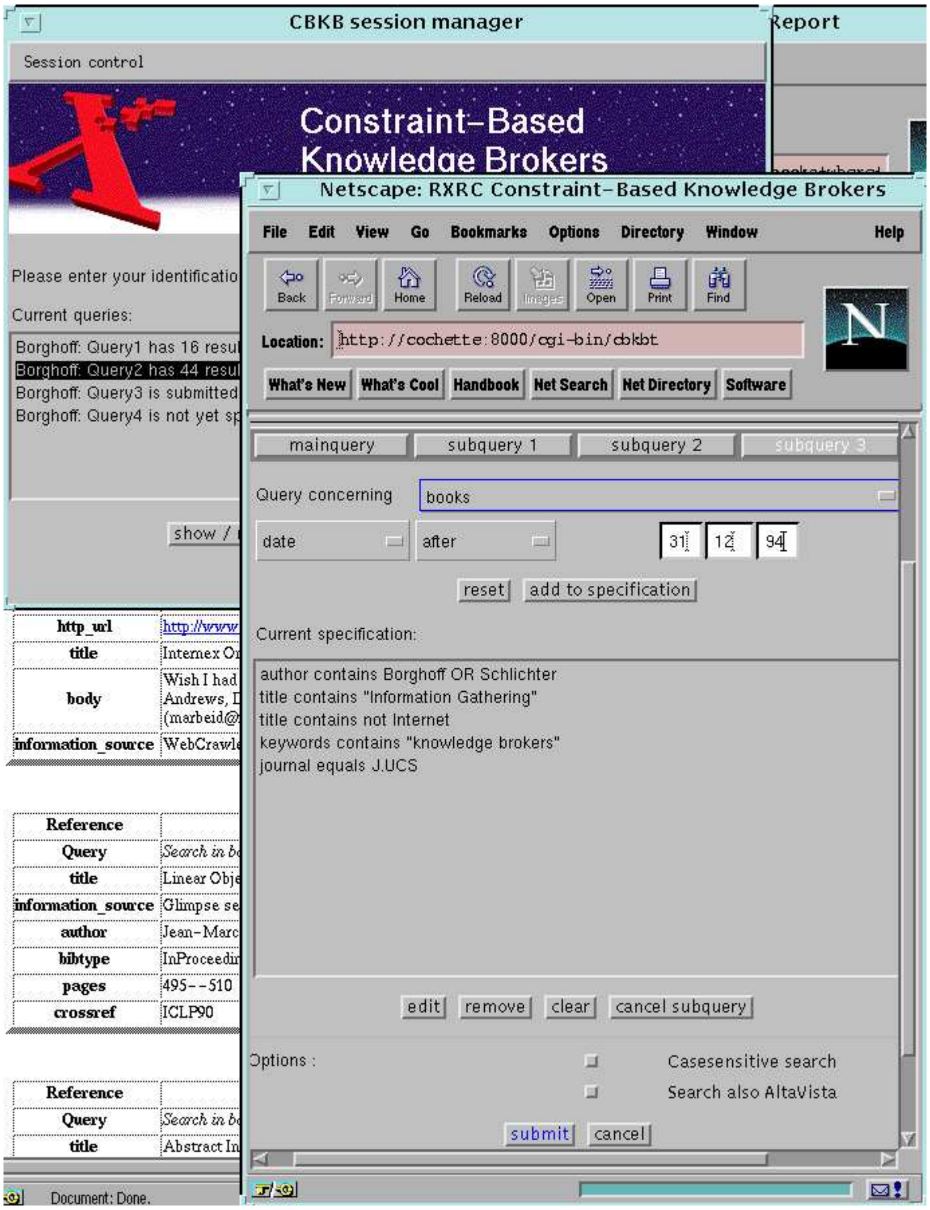


Figure 1: Architecture sketched with an example configuration

coordinators programmed in LO [Andreoli, 1995], while the user interface (programmed in Java, cf. Fig. 2) and the wrappers (programmed in Python) are external processes accessed through ForumTalk plugs.



This screenshot includes (from front to back) the *query window* where users formulate sophisticated queries, the *session control* where users organize parallel query sessions, and the *report window* where answers are converted to HTML for further processing.

Figure 2: Graphical user interface

The system is up and running. For more details on how to run the system, check http://www.xerox.fr/ and follow grenoble/ct/prototypes/cbkb/home.html.

3.3 Knowledge Brokers with Constraints Solvers

In the following we concentrate on a specific implementation of knowledge brokers, the so-called Constraint-Based Knowledge Brokers (CBKB) [Andreoli et al., 1996]. CBKB search heterogeneous information in various databases organized along different formats and with different requirements in mind. The basic model of retrieval exploits the standard request/subrequest model [Vielle, 1986] where subrequest generation depends on results obtained from precedent requests.

CBKB specify, in a single formalism, knowledge search requests, the answers to these requests, and the state of the knowledge brokers. This formalism extends the notion of standard feature constraints. The full formalism, including the state specification of the knowledge brokers, is not relevant for the remainder of the paper. The interested reader is referred to [Andreoli et al., 1995] for a detailed discussion.

In the most primitive reading, feature constraints are simple multisets of attribute/value pairs. Feature constraints have been extensively studied [Aït-Kaci et al., 1994] and, among other things, can be naturally exploited to provide structured representations of electronic information.

Assume we are interested in some recent publications about knowledge brokers written by the colleagues of a given person. First of all we have to find out who is a colleague. The Internet provides this information already for many company employees. For example, the electronic Xerox phonebook allows searches of the form "who is in the same group as ⟨person⟩". For all or some of the corresponding colleagues of *person*, we subsequently search for their recent publications about the given topic (in the example below the given topic is "knowledge brokers"). Again, there is a possibility to search for this kind of information. Among the variety of possible index/search subsystems, the Library of Congress WAIS-gateway or an installation of the Glimpse subsystem for bibliographical data fulfill our needs. Let us therefore assume the following query of a broker *broker_phone_glimpse* constraining a variable *R*.

```
<Id>
  R
  R: mainquery,
  R: subquery1 -> Pos1,
     Pos1: Person-Search,
     Pos1: pattern -> X1, X1: "who is in the same group
                              as Steve Freeman",
     Pos1: surname -> X2,
  R: subquery2 -> Pos2,
     Pos2: Publication-Search,
     Pos2: pattern1  -> X2,
     Pos2: pattern2  -> X3, X3: "knowledge brokers",
     Pos2: year    -> X4, X4 > X5, X5: 1992
```

Within a constraint, interdependencies or precedence conditions may exist between feature entries: in our example, the variables for *surname* and *pattern1* must coincide, the variable for *year* must hold a value greater than 1992.

The query is tagged with a unique identifier *Id*. The constraint variable *R* knows two request positions, one of type *Person-Search* (position *Pos1*) and one of type *Publication-Search* (position *Pos2*). The mediator of the wrapper component maps each type to the appropriate information source. In the case of our example, *Person-Search* will be mapped to the Xerox phonebook while the Glimpse subsystem is used for the publication search. Thus, the query of broker *broker_phone_glimpse* can be broken into two subqueries. The feature *pattern* specifies the search pattern for *Person-Search* while the combination of the features *pattern1* and *pattern2* define the pattern of the subsequent search within the information repository selected for *Publication-Search*.

It is worthwhile to note that a threshold mechanism [Andreoli et al., 1994] allows the withholding of the submission of a subquery until sufficient information is specified. In our case, assume that the subquery to *Publication-Search* needs at least the surname, i. e. a value for *X2*.

Therefore, the first subquery

```
<Id>
  Pos1
  Pos1: Person-Search,
  Pos1: pattern -> X1, X1: "who is in the same group
                            as Steve Freeman",
  Pos1: surname -> X2
```

is submitted to *broker_phone* to search in the people information repository.

Upon reception of an answer, *broker_phone* checks whether the answer is plausible. More specifically, a constraint solver checks whether the answer satisfies the property expressed in the query's constraint, i. e. whether the answer entails the request. A satisfactory answer is sent to the requester *broker_phone_glimpse*, e. g.

```
<Id>
  Pos1
  Pos1: Person-Search,
  Pos1: pattern    -> X1, X1: "who is in the same group
                               as Steve Freeman",
  Pos1: surname    -> X2, X2: "Andreoli",
  Pos1: given_name -> X6, X6: "Jean-Marc",
  Pos1: nickname   -> X7, X7: "JM"
```

Since the threshold is now reached, the second subquery to *broker_glimpse* is immediately launched as

```
<Id>
  Pos2
  Pos2: Publication-Search,
  Pos2: pattern1 -> X2, X2: "Andreoli"
  Pos2: pattern2 -> X3, X3: "knowledge brokers",
  Pos2: year   -> X4, X4 > X5, X5: 1992
```

The corresponding constraint specifies a query for the constrained variable *Pos2* of type *Publication-Search*. The constraint variable *Pos2* includes the form attributes *pattern1* and *pattern2* which are mapped by the mediator to appropriate search attributes of the selected bibliographical database. A *year*-attribute is required for each answer.

Again upon reception of an answer, *broker_glimpse* checks whether the answer entails the query. If so, the answer is sent to *broker_phone_glimpse*. In turn, the knowledge broker *broker_phone_glimpse* checks whether the full answer entails the original query. If so, we have obtained the first relevant publication of our interest.

3.4 Wrappers

A wrapper consists of a mediator and a set of database interfaces defining the interfaces to these external databases which may be accessed for knowledge retrieval. Mediator and database interfaces communicate through a constraint exchange protocol. They may reside on different machines connected by local or wide area networks.

The mediator acts as a client while the database interfaces play the role of servers. Thus, a database interface is continuously waiting for queries from the mediator, i. e. requests from knowledge brokers (in our example, from the knowledge brokers *broker_phone* and *broker_glimpse*). The protocol between the knowledge brokers and the mediator simply exchanges structured messages in the form of constraints.

There may exist multiple wrappers in the environment providing broker access to a wide variety of external databases. The scope of different wrappers may even overlap because duplicate and redundant information will be filtered out by the knowledge brokers [Andreoli et al., 1996].

3.4.1 Mediator

The mediator handles the communication between the knowledge brokers and the database interfaces. It accepts queries of knowledge brokers and returns to them answers received from the external databases via the database interfaces. The mediator uses the same internal format as the knowledge brokers (i. e. constraints) to represent queries and answers.

The mediator's main functionality is the selection of the information source to which a search request should be sent in order to satisfy the query. The initial constraint sent by a knowledge broker specifies the information source only according to the type of the requested information rather than by providing its name. The mediator determines the name of the appropriate information source and inserts it into the original query. In the subquery submitted by *broker_phone* the form attribute *Person-Search* is replaced by the database name *XeroxPhone*:

```
<Id>
  Pos1
  Pos1: XeroxPhone,
  Pos1: pattern -> X1, X1: "who is in the same group
                            as Steve Freeman",
  Pos1: surname -> X2
```

The database name also incorporates the network address, e. g. the URL in case of a Web-interface. For simplicity reasons that information is not explicitly represented in our example. The form attribute *pattern* need not be replaced because the Xerox phonebook uses the same attribute name to specify search pattern. The modified constraint is then propagated to the database interface which handles queries to the selected information source.

Currently, the system manages the mapping between the type of information source and its name and address by simple table lookups. In the next phase, sophisticated mediators [Wiederhold, 1992] will provide a more flexible mapping, similar to the mechanisms used in the UMDL-project [Birmingham, 1995; Birmingham et al., 1995]. There, so-called registry agents capture the address and content information of each database. Query planning agents negotiate with the registry agents about particular addresses of databases (see Fig. 3). During the negotiation phase, facilitator agents might be involved to resolve different protocols. [Borghoff et al., 1996a] discuss other ways of multi-agent coordination.

The mediator also handles the communication to the knowledge brokers. Answers received by the database interfaces are propagated to the correct knowledge broker determined by the identifier included in the answer.

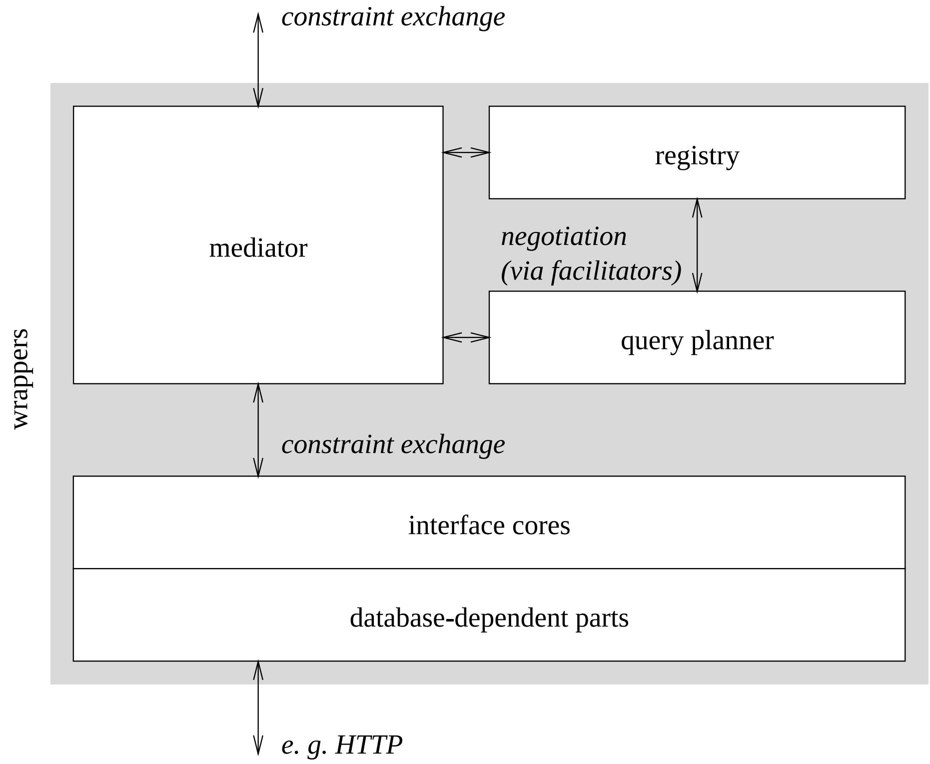


Figure 3: Negotiation between mediator, registry, and query planning agents

3.4.2 Database Interface

Our distinction between different database interfaces is based on the categorization of external databases according to their supported interfaces for submitting queries and returning answers. Examples of database categories are HTTP, Z39.50 and SQL. As illustrated in Fig. 1, each database interface itself consists of a database-independent (the interface core) and several database-dependent parts.

The *database-independent part* handles the receipt of queries, the construction of answers which are sent to the appropriate knowledge brokers via the mediator, and, in the case of differing internal formats used by knowledge brokers and the database interface, the conversion of queries and answers.

The following discussion is based on our implementation of the HTTP database interface. Other database interfaces are quite similar even if they might differ in their internal data formats.

Within the HTTP database interface queries and answers are represented by structured information consisting of an identifier, a database name, and multiple attribute/value pairs. In principle, the specification of the query and the answers managed by the database interface could use the same formalism as the knowledge brokers and the mediator. However, the specification of the database interface does not enforce constraint solving. Therefore, it is cumbersome to resolve interdependency relationships, and, in most cases, it is fully irrelevant to know of precedence conditions. Attribute/value pairs are easier to handle and convenient to implement. They control the query sent to external databases supporting the HTTP interface.

For example, the specification

```
<Id>
Glimpse
  query1 = "Andreoli"
  query2 = "knowledge brokers"
  errors = 0
  case   = "on"
```

results in a query sent to the Glimpse subsystem. Note here, that the form attributes *pattern1* and *pattern2* of the original request are mapped into the form attributes *query1* and *query2* used by the Glimpse interface. The values of *query1* and *query2* are combined and used as search patterns disallowing any error during case-insensitive pattern matching.

The interface core constructs out of the answers received from the external database, and parsed by the database-dependent part, answer messages represented in the internal format of the database interface. In the case of the HTTP interface each answer message consists of an identifier for matching the query and the answer messages, and a set of attribute/value pairs representing the information of a single search hit. In general, the attribute names are derived from the answers received from the external database.

For example, the query above may result in the following answer message.

```
<Id>
  bibtype   = "inproceedings"
  key       = "PASCO-1994"
```

```
author    = "J.-M. Andreoli and U. M. Borghoff and R. Pareschi"
title     = "Constraint-Based Knowledge Brokers"
booktitle = "Proc. 1st Int. Symp. on Parallel Symbolic
             Computation (PASCO'94)"
address   = "Hagenberg/Linz, Austria"
editor    = "H. Hong"
publisher = "Lect. Notes Series in Comp. Vol. 5. Singapore,
             New Jersey, London, Hong Kong: World Scientific"
year      = "1994"
month     = "sep"
pages     = "1--11"
url       = "http://www.xerox.fr/
             grenoble/ct/articles/94-pasco.ps.Z"
```

The interface core generates for every search hit received from the external database a separate answer message. These answer messages are then converted into constraints which are then sent to the mediator for propagation to the requesting knowledge broker.

The special characteristics of the individual external databases are encapsulated in the *database-dependent parts* which deal with the structure of the submitted queries as well as the structure of the answers received from the database.

For databases accessed over the Web, the structure of the queries are quite similar. Besides the database address the only additional difference is the number and names of the form attributes to be provided by the query. Typically, the knowledge brokers do not specify all attributes possible in the form. Some of the attributes are optional, or indeed unknown. If unspecified attributes are needed or useful during the query, the database-dependent part appends these attributes with default values. This explains the entries

```
errors = 0
case   = "on"
```

in the example above.

On the other hand, the knowledge brokers may have specified attributes that have no corresponding form attribute. Attributes that cannot be converted and assigned to an appropriate form entry, or attributes that have no specified value, are simply ignored for the query. This explains why the entry

```
year > 1992
```

is missing in the example above.

In general, it is necessary to examine the HTML form page manually in order to extract the required information for both the query composition as well as the answer decomposition.

However, the more complex problem is the answer decomposition, that is, how to extract automatically the relevant information representing answers to our previously submitted queries. The specification and the implementation of parsers interpreting database-specific answers require a lot more effort because the structure of database entries is not standardized. Especially in the Web, HTML pages are created by individual designers according to their own taste and perspective. In most cases these pages are prepared for visual presentation to the user and not to support automatic interpretation by agents. Examples of structuring methods for distinguishing between different search hits include definition lists, preformatted text, set of anchors where each anchor represents one search hit, and occasionally a list of entries separated by paragraph markers. For example, the Glimpse subsystem returns the entries of the bibliography databases as they were entered by the database administrators. Currently it is not possible to fully automate the generation of parsers which extract relevant information of answers and propagate them to the knowledge brokers. However, due to similarities of the answers a common base may be used for all database-dependent parts. Each database-dependent part must be manually customized and the effort depends on the complexity of the answer information provided by the external database.

Adding new external database interfaces to the prototype system is quite straight forward. First, a new database-dependent part must be provided. In many cases it can be derived from an existing one by integrating the appropriate modifications. Second, the new database-dependent part must be integrated into the interface core which handles the type of interface supported by the external database. Third, the dictionary of the mediator must be extended to include the new external database.

3.4.3 Conversion between Constraints and Database Interface Formats

The knowledge brokers as well as the interface core use a specific internal format to represent the query and the answer to the query. As we have seen, knowledge brokers use a specific format suitable to their constraint solvers whereas the interface core uses a data format specifically adapted to the functionality of the database interface protocol. For example, the HTTP interface core handles multiple attribute/value pairs (without interdependencies between attributes) convenient for implementation. Thus, every interface core which uses an internal data format different from constraints as used by knowledge brokers incorporates a converter between these two data formats. As already demonstrated above we will use the HTTP interface core as an example.

From a constraint specification to a specification using attribute/value pairs, the converter extracts first the identifier, second, using the constrained variable and its type, the database name, and then the feature entries with a possibly referenced value.

From a specification using attribute/value pairs to a constraint specification, the obvious task of the converter is to instantiate the free variables in the query's constraint with received values (in the current prototype, a first soundness check is done within the converter itself).

When the interface core provides additional information within the answer, the converter generates new feature entries extending the original constraint of the query. In the example above, additional feature entries for the given name and the nickname were added. Subsequent searches may use the given name to further prune the search specification or to extract the publication of the wanted person, if the surname is ambiguous. The same holds true for the additional entry *url*. As Fig. 1 (see dashed arrows) illustrates, a knowledge broker *broker_document* may use this anchor for a subsequent search to retrieve the postscript version of the paper in question.

4 Shortcomings of the Current Approach

Knowledge brokers must cope with an open, heterogeneous world where information repositories provide a wide variety of different query and answer formats. As already mentioned in Sect. 3.4.2, database-dependent parts must be customized manually because of the heterogeneity with respect to the structuring of the answers as well as the lack of any information describing the semantics of answer components. However, it would facilitate the integration of new database-dependent parts if external databases would support empty queries which return as results the structure of accepted queries as well as the structure of returned answers. That information could be used to automatically generate the parser for interpreting answers and the code for constructing queries. Within the broker environment there are still conventions necessary which match the information received from the mediator to the appropriate attributes derived from the empty query. However, these conventions would be local to the broker environment and would not require external information repositories to follow these standards. Tsimmis [Chawathe et al., 1994] also attempts to apply a generator approach to integrate new information sources into their environment. There, a human might study samples of the data and develop an appropriate specification for the interface generator. Harvest pursues an object-oriented approach to enable the representation, manipulation and display of arbitrarily complex data in application-specific ways [Chhabra et al., 1994]. An access of a Web datum results in the invocation of the appropriate method which is stored in a type/method registry. Methods are executed either locally or remotely.

Another shortcoming is caused by the type and format of the information provided by the external databases. Often they do not provide the information in a way necessary to interoperate efficiently with other databases in subsequent searches. For example, queries to the Xerox phonebook return fully specified names of people including nicknames, given names, initials, surname, and generation qualifiers, such as "jr" or "III". Using for instance the name "Jimmy Jim R. Smith jr" without any modifications as pattern for a subsequent search in a bibliographical database will rather likely result in no or very few answers. The initial query to the Xerox phonebook leads to overspecified search patterns for subsequent searches in bibliographical databases. Thus, it is necessary to extract the relevant information which in most cases is specific to the external database and often even to the individual answer.

As already known from traditional ways of accessing large information repositories there is often a thin line between over- and underspecified queries. However, generally the user can refine the constraints in subsequent queries to the same database in order to control the amount of information returned. In the case of complex broker systems this is rather difficult because the answers of multiple databases are combined by the brokers to construct new queries to other databases. Currently the answers do not include enough semantics to extract automatically that information which is relevant for subsequent queries. That information must be provided manually by the designer of the database-dependent part. Ideally, all backends would encapsulate a query-customizable extraction method within the answers. For a first step towards this vision see [Schwartz and Hardy, 1996].

A related problem exists for complex queries involving multiple sources of information. The decomposition into individual queries must be implemented manually to cater subsequent individual searches. The constraint-based approach, however, already paves the way for sophisticated expressions of data dependencies.

An issue only briefly mentioned so far is the mechanism for localization of external databases. For the current prototype system all external databases were localized manually and incorporated into the mediator dictionary individually. This approach is not feasible for the rapidly growing number of information repositories. Thus, the networked community must provide directory services which associate information types or subject areas with database names, location information and methods to access the database entries. White pages are not sufficient; we need yellow pages equipped with meta-information of semantical behavior. The directory information could be constructed passively using indexing tools such as Archie or the World-Wide Web Worm, or actively by requiring information repositories to register their database services. In the former case only a limited amount of information can be extracted automatically by these tools. In the latter case information repositories could provide a lot more semantical information to the directory services. This meta-information could then be used by mediators and knowledge brokers to select the appropriate external databases for knowledge retrieval and to adjust to the query decompositions. Again, mediators can support the query process effectively. For instance, mediators may select, with respect to the requester site, the best replica of the external database out of a set of mirror sites. On the other hand, they may choose, with respect to some ranking process *à la* gGLOSS [Gravano and Garcia-Molina, 1995], among different databases providing related information.

5 Conclusion

Due to the proliferation of heterogeneous information sources and their availability over the Internet, we aimed at searching these sources within a uniform framework for knowledge brokers. We allowed the combination of search results, the extraction of knowledge from one search in order to trigger a subsequent search, and, finally, the evaluation of the hits according to constraints.

We introduced the overall architecture and discussed the main constituents such as user interface, knowledge brokers, wrappers, and external databases. Knowledge brokers specify electronic information through constraints, whereas wrappers, providing the interfaces to existing external databases, specify the information in a database convenient format, e. g. attribute/value pairs. A converter translates between both representations. We explained the current state of a prototype system that distinguishes between a mediator, a database-independent part (the interface core) and specific database interfaces.

We finally discussed shortcomings of the current approach. These are mostly caused by the emphasis of current information repositories and their interfaces, above all their poor support for knowledge combination and the difficulty of localizing the appropriate information repositories.

Despite the pragmatism to stick to essentials while implementing a prototype for an early proof of concept, the current approach (with one wrapper containing a dozen database interfaces) performs well. The adaptation of additional database interfaces to new data sources is straight forward. In contrast to the time that was needed to implement the database-independent part as well as the knowledge brokers, the time to implement an additional new database interface is negligible. For instance, the database-dependent part of the Xerox phonebook interface was written in three hours, and of the Glimpse interface in less than a day.

References

[Aït-Kaci et al., 1994] H. Aït-Kaci, A. Podelski, and G. Smolka. A feature-based constraint-system for logic programming with entailment. Theoretical Computer Science, 122, 263–283, 1994

[Aliweb, URL] http://web.nexor.co.uk/public/aliweb/aliweb.html

[Alta Vista, URL] http://altavista.digital.com/

[Andreoli et al., 1994] J.-M. Andreoli, U. M. Borghoff, and R. Pareschi. Constraint-based knowledge brokers. In H. Hong, editor, Proc. 1st Int. Symp. on Parallel Symbolic Computation (PASCO '94), pp. 1–11, Hagenberg/Linz, Austria, September 1994. Lecture Notes Series in Computing **5**, Singapore, New Jersey, London, Hong Kong: World Scientific

[Andreoli et al., 1995] J.-M. Andreoli, U. M. Borghoff, R. Pareschi, and J. H. Schlichter. Constraint agents for the information age. J. Universal Computer Science, 1, 12, 762–789, December 1995. Electronic version available at http://hyperg.iicm.tu-graz.ac.at/Cjucs_root

[Andreoli et al., 1996] J.-M. Andreoli, U. M. Borghoff, and R. Pareschi. The constraint-based knowledge broker model: Semantics, implementation and analysis. J. Symbolic Computation, 1996 (in press)

[Andreoli, 1995] J-M. Andreoli. Programming in ForumTalk. Technical Report CT–003, Rank Xerox Research Centre, Grenoble Lab., France, 1995

[Arcelli et al., 1995] F. Arcelli, U. M. Borghoff, F. Formato, and R. Pareschi. Tuning constraint-based communication in distributed problem solving. In Proc. 1st Int. Workshop on Concurrent Constraint Programming (CCP '95), Venice, Italy, May 1995

[Barbara and Clifton, 1992] D. Barbara and C. Clifton. Information brokers: Sharing knowledge in a heterogeneous distributed system. Technical Report MITL–TR–31–92, Matsushita Information Technology Lab., Princeton, NJ, October 1992

[Barbara, 1993] D. Barbara. Extending the scope of database systems. Technical Report MITL–TR–44–93, Matsushita Information Technology Lab., Princeton, NJ, 1993

[Birmingham et al., 1995] W. P. Birmingham, E. H. Durfee, T. Mullen, and M. P. Wellman. The distributed agent architecture of the University of Michigan digital library. In Proc. AAAI Spring Symp. Series on Information Gathering from Distributed Heterogeneous Environments, Stanford, CA, March 1995

[Birmingham, 1995] W. P. Birmingham. An agent-based architecture for digital libraries. D-Lib Magazine, July 1995. Available at http://www.cnri.reston.va.us/-home/dlib/July95/07contents.html

[Borghoff et al., 1996a] U. M. Borghoff, P. Bottoni, P. Mussio, and R. Pareschi. A systemic metaphor of multi-agent coordination in living systems. In A. Javor, A. Lehmann, and I. Molnar, editors, Proc. Europ. Simulation Multiconf. (ESM '96), pp. 245–253, Budapest, Hungary, June 1996. San Diego, CA: The Society of Computer Simulation

[Borghoff et al., 1996b] U. M. Borghoff, R. Pareschi, F. Arcelli, and F. Formato. Constraint-based protocols for distributed problem solving. Science of Computer Programming, 1996 (in press)

[Borghoff et al., 1996c] U. M. Borghoff, R. Pareschi, H. Karch, M. Nöhmeier, and J. H. Schlichter. Constraint-based information gathering for a network publication system. In Proc. 1st Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96), pp. 45–59, London, U. K., April 1996. Blackpool, U. K.: The Practical Application Company Ltd

[Bowman et al., 1994a] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The Harvest information discovery and access system. In Proc. 2nd Int. World-Wide Web Conf., pp. 763–771, Chicago, IL, October 1994

[Bowman et al., 1994b] C. M. Bowman, C. Dharap, M. Baruah, B. Camargo, and S. Potti. A file system for information management. In Proc. Conf. on Intelligent Information Management Systems, Washington, DC, June 1994

[CACM, 1994] Special issue on intelligent agents. Communications of the ACM, July 1994

[Callan et al., 1992] J. P. Callan, W. B. Croft, and S. M. Harding. The Inquery retrieval system. In Proc. 3rd Int. Conf. on Database and Expert Systems Applications, pp. 78–83, September 1992

[Callan et al., 1995] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In Proc. 18th ACM SIGIR Conf. on Research and Development in Information Retrieval, Seattle, WA, July 1995

[Chawathe et al., 1994] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The Tsimmis project: Integration of heterogeneous information sources. In Proc. IPSJ Conf., Tokyo, Japan, October 1994. Los Alamitos, CA: IEEE Comp. Soc. Press

[Chhabra et al., 1994] B. Chhabra, D. Hardy, A. Hundhausen, D. Merkel, J. Noble, and M. F. Schwartz. Integrating complex data access methods into the Mosaic/WWW environment. In Proc. 2nd Int. World-Wide Web Conf., pp. 909–919, Chicago, IL, October 1994

[CS_biblio, URL] http://liinwww.ira.uka.de/bibliography/index.html

[Denzinger, 1995] J. Denzinger. Knowledge-based distributed search using Teamwork. In V. Lesser, editor, Proc. 1st Int. Conf. on Multi-Agent Systems (ICMAS '95), pp. 81–88, San Francisco, CA, June 1995. Menlo Park, Cambridge, London: AAAI Press, MIT Press

[Emtage and Deutsch, 1992] A. Emtage and P. Deutsch. Archie: An electronic directory service for the internet. In Proc. Usenix Conf. Winter '92, pp. 93–110, Sunset Beach, CA, January 1992. Berkeley, CA: Usenix Association

[Fikes et al., 1995] R. Fikes, R. Engelmore, A. Farquhar, and W. Pratt. Network-based information brokers. In Proc. AAAI Spring Symp. Series on Information Gathering from Distributed Heterogeneous Environments, Stanford, CA, March 1995

[Glimpse, URL] http://glimpse.cs.arizona.edu:1994/bib/

[Gravano and Garcia-Molina, 1995] L. Gravano and H. Garcia-Molina. Generalizing gloss to vector-space databases and broker hierarchies. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, Proc. 21st Int. Conf. on Very Large Data Bases, pp. 78–89, Zurich, Switzerland, September 1995. San Francisco, CA: Morgan Kaufmann

[Harvest, URL] http://harvest.cs.colorado.edu/

[Kahle and Medlar, 1991] B. Kahle and A. Medlar. An information system for corporate users: Wide area information servers. Connexions: The Interoperability Report, 5, 11, 2–9, November 1991

[Lycos, URL] http://lycos.cs.cmu.edu/

[Manber and Wu, 1994] U. Manber and S. Wu. Glimpse: A tool to search trough entire file systems. In Proc. Usenix Conf. Winter '94, pp. 23–32, San Francisco, CA, January 1994. Berkeley, CA: Usenix Association

[Mauldin and Leavitt, 1994] M. L. Mauldin and J. R. R. Leavitt. Web agent related research at the center for machine translation. In Proc. ACM Special Interest Group on Networked Information Discovery and Retrieval (SIGNIDR'94), McLean, VA, August 1994

[McBryan, 1994] O. A. McBryan. GENVL and WWWW: Tools for taming the web. In O. Nierstrasz, editor, Proc. 1st Int. World-Wide Web Conf., Geneva, Switzerland, May 1994

[Obraczka et al., 1993] K. Obraczka, P. B. Danzig, and S.-H. Li. Internet resource discovery services. IEEE Computer, 26, 9, 8–22, September 1993

[Schwartz and Hardy, 1996] M. F. Schwartz and D. R. Hardy. Customized information extraction as a basis for resource discovery. ACM Transactions on Computer Systems, 14, 2, May 1996

[Schwartz et al., 1992] M. F. Schwartz, A. Emtage, B. Kahle, and B. C. Neuman. A comparison of internet resource discovery approaches. Computing Systems, 5, 4, 461–493, Fall 1992

[Sheth and Larson, 1990] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 22, 3, 183–236, September 1990

[Veronica, URL] gopher://gopher.unr.edu/11/veronica

[Vielle, 1986] L. Vielle. Recursive axioms in deductive databases: The query-subquery approach. In L. Kerschberg, editor, Proc. 1st Conf. on Expert Database Systems, Menlo Park, CA, April 1986. Benjamin/Cummings Publ. Company

[Wiederhold, 1992] G. Wiederhold. Mediators in the architecture of future information systems. IEEE Computer, 25, 3, 38–49, March 1992

[Wooldridge and Jennings, 1995] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. Knowledge Engineering Review, 10, 2, 115–152, 1995

[WWWW, URL] http://wwww.cs.colorado.edu/WWWW/

Acknowledgement

The system described in this paper is an effort of many people. We would like to especially thank Jean-Marc Andreoli, Pierre-Yves Chevalier, Martina Nöhmeier (visiting from TU München, Germany), Remo Pareschi, and Jutta Willamowski. We thank Francesca Arcelli and Ferrante Formato (both Univ. Salerno, Italy) for their work on theoretical aspects of information reuse during the brokerage sessions. For the integration of the system into the so-called *Physicists Network Publishing System* thanks are due to Bernd Diekmann, Eberhard Hilf, Thomas Severiens, Heinrich Stamerjohanns (all Univ. of Oldenburg, Germany), and Harald Karch (Rank Xerox Germany). Last but not least we would like to thank the anonymous referees for their thorough comments which greatly helped to improve the paper.