SemaLink: An Approach for Semantic Browsing through Large Distributed Document Spaces

S. Wiesener¹, W. Kowarschick², R. Bayer¹,²

¹Bayerisches Forschungszentrum für Wissensbasierte Systeme (FORWISS), München
²Fakultät für Informatik, Technische Universität München
email: [wiesener, kowa, bayer]@informatik.tu-muenchen.de

Abstract
Hypermedia retrieval in combination with querying is a very powerful document access metaphor for digital library systems. Conventional distributed hypermedia systems, however, suffer from some well-known problems that make effective, goal-directed document retrieval and maintenance impossible. On the one hand, relationships between documents are modeled on a very low level of abstraction, preventing knowledge re-use and user-adapted navigation through the document space. On the other hand, huge static web structures are prone to redundancies, inconsistencies and costly to maintain.

In this paper we present an approach of a scalable distributed resource discovery and delivery system that offers access to information through querying and knowledge-based hypermedia browsing. Relationship knowledge is partitioned into handy, independent structures, modeled on a high level of abstraction apart from documents. Navigation paths are computed by the system, combining semantic networks with information retrieval. A theoretical framework as well as implementation issues are discussed.

Keywords
digital libraries, hypermedia, resource discovery, information retrieval, knowledge-bases

1 Motivation
Actually more than 40 million participants are estimated to have access to the Internet worldwide, many of them being not only passive consumers but also qualified information providers. Electronic publishing in the Internet became an integral part of scientific information exchange, though there is no homogeneous content-based access structure to information resources. Since the bandwidth and global development of wide area networks is increasing continuously the publishing process is becoming more and more decentralized, often down to the level of single authors. As a direct consequence, the information resource discovery problem has reached a new dimension.

While, on the one hand, the Internet chaos may be exciting and enriching, it, on the other hand, causes inefficiency when goal directed information resources are needed to solve problems. Nothing seems to have changed: today even more short-lived insider experience is needed to find resources in our networks than it was in the traditional libraries with miles of dusty bookshelves. The technical infrastructure to precisely access any single document is available, but cannot be used effectively. The reason for that is the low level of knowledge disclosed to the information systems themselves, which is insufficient to enable intelligent information routing and user support. Evolving from traditional publishing methods, digital documents today still hide most of the authors’ knowledge and competence in a natural language and layout structure. Disclosing more meta-data would allow digital library systems to offer a semantic and therefore goal-directed navigation through large document spaces.

Each single publication contributes as a small piece to a mosaic of individual readers’ interests. The value of a document is determined by its contents on the one hand and its user specific relevance [13], i.e., its right places in such mosaics on the other hand. Therefore, it is not only
important to provide documents but also knowledge that helps to find right places for information resources in those mosaics. It does not matter whether such knowledge-bases are built up manually or derived automatically from other sources, though.

The process of human information search usually consists of one or more search steps. A user who has a complex non-standard problem to solve does not know in advance what pieces of information are necessary to find an appropriate solution. He will therefore not be able to formulate precise queries that yield exactly the desired information. Rather the user will iteratively try to locate documents by means of broad queries and from there on try to refine his knowledge by following cross-references that lead to related information.

The search process consists of querying and browsing steps involving two different sorts of meta-knowledge that support the user in finding information [9]. Firstly, documents should be described by a set of attributes that allows a direct access to them. Typical attributes suitable for document retrieval are Title, Author, Keywords, Classification, Mediatype etc. Even full-text retrieval can be seen as an attribute search where all occurring words are stored in one attribute. Classical digital library and catalog systems [1], [2], [10] offer attribute retrieval only.

Secondly, relationships between documents have to be maintained in the system to enable browsing. Such a relationship consists of at least a source description and a destination description of related documents. In addition, it may contain further attributes like link type or weight [7].

![Figure 1: A virtual hyperlink passing two knowledge-bases](image)

In most hypermedia systems (e.g. Hyper-G [8] or World-Wide Web [4]) hyperlinks are modeled on a very low level of abstraction. They explicitly point from one chunk of information to another. This aggravates the well-known lost-in-hyperspace syndrome in large-scale systems. The intentions, background-knowledge and context the authors had at link creation time usually do not match with those that individual users have during retrieval. Users are therefore condemned to follow a lot of links pointing to non-relevant information. Even if a link is appropriate to the user’s context, it does not guarantee to deliver all related documents that might be relevant in that context. So hypermedia retrieval can degenerate into a mixture of depth-first and breadth-first docuverse exploration soon overcharging the user’s cognitive abilities. Besides, the consistent administration of global hypermedia webs is costly, if not completely impossible.
To overcome those problems and to achieve a system-supported goal-directed navigation through the document space two fundamental changes are necessary. At first, knowledge about relationships between documents has to be modeled on a higher level of abstraction. That means, source and destination descriptors of hyperlinks should, in general, not address information by means of explicit pointers. Rather descriptors should contain attributes that describe related documents in a declarative manner. Secondly, authoring, maintaining, storing, and distribution of relationship knowledge should be done independently from documents. This is possible, on principle, because one does not necessarily have to be a document author to formulate relationships between topics or concepts.

The main idea of the SemaLink approach is that authors produce both documents and/or relationship knowledge [14]. Relationship knowledge interconnects different but related concepts. It is modeled in many independent, delimited semantic networks [12]. Nodes of these networks represent concepts, and typed and weighted edges express relationships between them. The whole distributed hypermedia is no longer a statically connected graph but a set of documents and rather small semantic networks. Documents and semantic networks now form a virtual hypermedia graph structure (see Figure 1), virtual in the sense that relationships between documents are partly computed by the system. They may cross several independent semantic networks. The joining of documents and semantic networks is achieved by means of information retrieval: Similarities amongst attributes of semantic nodes and documents are computed. Furthermore, the whole process can be supported by intelligent deductive retrieval clients that adapt the virtual graph structure according to a user model.

In contrast to conventional distributed hypermedia systems the SemaLink approach is not based on resource locators (e.g., URLs of the WWW [3]) only. This raises a technical resource discovery problem. Documents and relationship knowledge have to be located in a large unstructured information space before being combined in the above described manner.

In Section 2 a short framework for document publishing and retrieval in such a virtual distributed hypermedia structure is introduced. In Section 3 implementation issues, especially resource discovery problems are addressed. A combined resource discovery and delivery architecture is presented which shall make arbitrary documents and semantic networks available at user retrieval clients. Section 4 closes with summary and outlook.

2 A Distributed 2-level Hypermedia Framework

Although our approach is not based on a statically connected hypermedia web, the basic components remain nodes and links. We distinguish document nodes from semantic nodes. While document nodes contain multimedia data, semantic nodes are used to model semantic knowledge only. Links are static, interconnect semantic nodes and describe relationships between them. Document nodes, on the other hand, are connected to semantic nodes by means of similarities computed by the system. Furthermore, independent semantic networks may be connected in the same way. Thus, relationship knowledge between documents is formulated on a high level of abstraction. Navigation paths are not only determined by static links but also by virtual edges that represent similarities (see Figure 2). Note that in the SemaLink approach it is not necessary to explicitly link any new document to all related ones that are already part of the document space. Rather, existing knowledge is used to compute appropriate links dynamically.
2.1 Nodes and Links

Two object types accommodate all meta-data and media data that can be published by authors and retrieved by system users: node and link objects. For reasons of simplification we do not deal with composite nodes in this paper. However, extensions to handle complex nodes could be easily integrated.

Node objects (or nodes) represent multimedia data and descriptions of documents’ content. They consist of node contents and a node descriptor (cp. Dexter Reference Model [7]). The node contents contain arbitrary multimedia data and may be empty. Their internal structure is not known to the retrieval system, only to media presentation tools that are part of the retrieval client software. Their meaning is disclosed to users by viewing its presentation. Though node contents do not directly contribute to system supported querying and navigation they, of course, play an essential role in the whole search process. Users base their decisions for further search actions on the documents read so far.

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DRL (Descriptor Resource Locator)</strong></td>
<td>Unique resource locator identifying the node descriptor</td>
</tr>
<tr>
<td><strong>CRL (Contents Resource Locator)</strong></td>
<td>Unique resource locator identifying the node contents</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Data type of the node contents (e.g., text, video, audio, semantic)</td>
</tr>
<tr>
<td><strong>Authors</strong></td>
<td>Creators of the node</td>
</tr>
</tbody>
</table>

Table 1: Node object descriptor attributes
Node descriptors classify nodes by a set of attributes. Table 1 depicts a selection of possible attribute types and their meaning. Node descriptors have two main purposes. Firstly, they provide means to declaratively access node objects by querying their attributes without knowing their location. This is used for user query processing as well as for determining similar node objects to a given one. Secondly, node descriptors, of course, serve as additional meta-information to be displayed to users during node representation. Node objects with empty node contents (attribute type semantic) play a special role. They represent pure semantic information on an abstract level and are elements of semantic networks. These networks are the basis for browsing the virtual global web structure.

Link objects (or links) represent weighted relationships between semantic nodes. They consist of a set of attributes (see Table 1). Links are modeled and managed separately from nodes as nodes may exist independently from links and may be referenced by several link objects each.

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Organization</strong></td>
<td>Organization providing the node (e.g., research center or company)</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>Time of last update</td>
</tr>
<tr>
<td><strong>Title</strong></td>
<td>Title of the node</td>
</tr>
<tr>
<td><strong>Keywords</strong></td>
<td>Set of keywords describing the node contents</td>
</tr>
<tr>
<td><strong>Abstract</strong></td>
<td>Short textual description of the node contents</td>
</tr>
<tr>
<td><strong>Classification</strong></td>
<td>According to a classification scheme (e.g. ACM Computing Reviews Classification)</td>
</tr>
<tr>
<td><strong>Target_Group</strong></td>
<td>Classification of users (e.g., lay, advanced, professional)</td>
</tr>
</tbody>
</table>

Table 1: Node object descriptor attributes

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LRL (Link Resource Locator)</strong></td>
<td>Unique resource locator identifying the link</td>
</tr>
<tr>
<td><strong>Source_DRL</strong></td>
<td>Unique resource locator for identifying the source node descriptor</td>
</tr>
<tr>
<td><strong>Destination_DRL</strong></td>
<td>Unique resource locator for identifying the destination node descriptor</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Semantics of the link (e.g., is-a, part-of, has-prerequisite, opposes, supports)</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>Strength of the relationship (real number between 0 and 1)</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Creator of link</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>Time of last update</td>
</tr>
</tbody>
</table>

Table 2: Link object attributes
2.2 Navigation Paths

The hypertext interface metaphor can now be applied to navigate the distributed set of node and link objects. As described in Section 1 human information retrieval consists of search and browsing steps. Initially users locate start nodes by means of declarative queries. The system returns nodes satisfying these condition to a certain degree. These nodes may be documents with multimedia contents as well as semantic nodes with empty contents. Of course, it increases retrieval quality if query terms are weighted and the matching of query and node attribute terms is fuzzy. Result node descriptors are presented to the user in a list ranked according to their degree of similarity to the query. The user now decides which node contents are to be presented.

By path we denote a sequence of nodes interconnected by links and virtual edges. Virtual edge objects (or virtual edges) express computed similarities between distinct node objects, i.e., they are generated by the system, not created by authors. By now we do not care when virtual edges are created. Rather, we assume that they simply exist and may be accessed like link objects. It is necessary to model virtual edges on the level of single attributes, i.e., similarities of nodes depend on the attributes users want the system to consider. By this, users can choose for instance the attributes Keywords and Organization to derive relationships to documents without being bound to a certain Author (or vice-versa). A list of attributes of a virtual edge is depicted in Table 1.

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRL (Virtual Edge Resource Locator)</td>
<td>Identifier of virtual edge</td>
</tr>
<tr>
<td>DRL₁</td>
<td>Identifier of a node descriptor</td>
</tr>
<tr>
<td>DRL₂</td>
<td>Identifier of a similar node descriptor</td>
</tr>
<tr>
<td>SimAttr</td>
<td>Extent of similarity of attribute <code>attr</code> between the two nodes DRL₁ and DRL₂</td>
</tr>
</tbody>
</table>

Table 3: Virtual edge object attributes

Different types of attributes require different methods to derive similarity. For example, the similarity $\text{Sim}_{\text{Keywords}}(v)$ of a virtual edge $v$ may be expressed by the following formula

$$\text{Sim}_{\text{Keywords}}(v) = \frac{|K_1 \cap K_2|}{\min(|K_1|, |K_2|)}$$

where $K_1$ is the set of keywords of the node $DRL_1$ of the virtual edge $v$. This formula does not undervalue nodes that provide fewer keywords. Especially semantic nodes provide rather few keywords, as they represent abstract concepts.

An actually presented node can be the initial point for a knowledge-based browsing through the distributed information space. A navigation path is a sequence of virtual edges representing similarities and static links interconnecting semantic nodes. Figure 2 shows a navigation example. Links and virtual edges are labeled with their weight and similarity values. As criterion for similarity only the descriptor attribute Keywords is considered here. Author A published a paper that deals with Pentium motherboards. By using semantic knowledge of two independent authors B and C it can be derived that a motherboard has as a subcomponent a processor chipset whose 2nd-level cache may fulfill the COAST standard. So appropriate papers on COAST
cache memory may be located. Authors A and D possibly do not know each other and knowledge authors B and C perhaps never published a document or they created semantic networks primarily to interconnect their own documents. Their knowledge may so be reused in a great variety of retrieval contexts.

2.3 Deductive Path Determination

If path traversal shall be supported by a rule-based system, relevance has to be defined to restrict the navigation space. To this end paths have to be rated and ranked. Only deduced paths with a certain relevance are considered for further navigation. The relevance of a path depends on the following features:

- **Weight of link objects**: Authors can express the strength of a single relationship in an attribute Weight.

- **Similarity value of virtual edges**: Paths containing virtual edges with a low similarity value lose relevance because coherence and context are no more guaranteed.

- **Path length**: With increasing path length coherence decreases even if all edges are heavily weighted. For example, if a user is looking at a document that deals with cars in a very general way, a system-generated link should not directly point to the technical description of a sparking plug, even if there is a strong transitive is-part-of relationship.

On this basis the relevance \( \mathcal{R}_P \) of a path \( P \) can be expressed by the following formula

\[
\mathcal{R}_P = \prod_{v \in P} u_S(Sim(v)) \cdot \prod_{l \in P} u_W(Weight(l)) \frac{1}{u_L(L_P)}
\]

where \( v \) and \( l \) are virtual edges and links, resp. of the path. \( L_P \) is the number of links in the path. \( Sim(v) \) is a function that computes a similarity value between two nodes in the range [0,1] by means of appropriate functions \( Sim_{Attr}(v) \) that describe similarities between single attribute values. \( Weight(l) \) is the weight associated with link \( l \).

![Figure 3: Indexing functions](image)

The relevance calculation is adapted to individual users’ profiles by the indexing functions \( u_S \), \( u_W \), and \( u_L \) (see Figure 3 for examples). These indexing functions are maintained by the retrieval client for each single user. The function \( u_S \) expresses, e.g., that the user accepts already rather low similarities between node descriptors as good values. A similarity value of 0.5 yields
an index value of nearly 0.8. The function $u_w$, on the other hand, weakens links weighted with values less than 0.5 and favors weights above. Finally, $u_L$ determines how the actual pathlength is scored. In this example, the total relevance $\Re_p$ of a path is always less than 0.5 if the length exceeds 4 links.

Now let $\Re_{\text{min}}$ be the lowest bound for path relevance values the user will accept. The rules of the following logic program derive paths that have a relevance value $\Re_p \geq \Re_{\text{min}}$. $\Re_{\text{min}}$ is a constant that is part of the user model.

The predicate `relevant` computes an actual relevance value $\Re_p$ of a path and tests whether $\Re_p \geq \Re_{\text{min}}$ holds. Its argument $WS$ denotes the product of all contributing link weights and similarities of virtual edges. The argument $Length$ denotes the pathlength.

$$\text{relevant} (WS, Length) :\equiv \frac{WS}{u_L(Length)} \geq \Re_{\text{min}}.$$  

The following rule produces relevant paths of length 1. Link objects are represented by the predicate `link`. The arguments $N1$ and $N2$ denote nodes (i.e., DRLs) connected by the link. $Type$ is the link type, $Weight$ denotes the given weight, and $WS$ denotes the indexed weight of the path.

$$\text{edge} (N1, N2, Type, WS, 1) :\equiv$$  
$$\text{link} (N1, N2, Type, Weight),$$  
$$WS = u_w(Weight),$$  
$$\text{relevant} (WS, 1).$$  

The following rule maps virtual edges onto edges of length 0 in the same way. Virtual edges do not have influence on the pathlength.

$$\text{edge} (N1, N3, Type, WS, 0) :\equiv$$  
$$\text{virtual_edge} (N1, N2, Sim),$$  
$$WS = WS2 \cdot u_s(Sim),$$  
$$\text{relevant} (WS, 0).$$

Every single edge contributes to a corresponding path.

$$\text{path} (N1, N2, Type, WS, Length) :\equiv$$  
$$\text{edge} (N1, N2, Type, WS, Length).$$

While the above rules are basic elements of the rule-based navigation system the following rule is user specific. By such rules users determine how transitive relationships between nodes are built. Generally, many of them are part of a user model. $N1, N2,$ and $N3$ denote node objects. $WS, WS1,$ and $WS2$ denote the products of all weights and similarities belonging to the paths. $Length, L1,$ and $L2$ are the corresponding pathlengths.

$$\text{path} (N1, N3, \text{supported-by}, WS, Length) :\equiv$$  
$$\text{path} (N1, N2, \text{has-part}, WS1, L1),$$  
$$\text{edge} (N2, N3, \text{supported-by}, WS2, L2),$$  
$$WS = WS1 \cdot WS2,$$  
$$Length = L1 + L2,$$  
$$\text{relevant} (WS, Length).$$
This rule expresses that a path typed has-part connected to another path typed supported-by yields a new path typed supported-by. Note that has-part and supported-by are constants. The relevance of that path results from the product of all contributing weights and similarities of both original paths and their added up lengths. Of course, a new path will only be derived by that rule if it is relevant.

Observe that the presented example rules only consider local information at each deduction step. That means, information is not passed from node to node along the path. To avoid possible losses of coherence information passing should be possible on the level of knowledge authoring (i.e., when authoring node and link objects) as well as on the level of deduction rules. Moreover, no influences between derived paths are taken into consideration here. For instance, two different paths leading from one document to another should increase the relevance of the target document. Those problems are dealt with in our current work.

3 Implementation Issues

The special characteristics of SemaLink objects have big influence on a system implementation. Authors create node and link objects at their local hosts. As the node contents may contain arbitrary multimedia information their storage and network transmission is rather expensive [11], [5]. To avoid unnecessary replication and transmission node contents therefore remain stored at the author’s local host and are accessed from there only if a node is to be presented to the user. This already guarantees a certain degree of scalability. Node descriptors and link objects, on the other hand, contain nothing but textual attributes which usually make up a fraction of the corresponding node contents size only. Thus node descriptors and link objects may be replicated and transmitted whenever this significantly increases retrieval quality. By means of node descriptors and link objects, authors provide all information to the system needed for direct document access and path deduction.
A remaining problem are virtual edges. In contrast to static links they are not equipped by authors with the resource locators of the corresponding nodes they interconnect. Rather, they are produced from the system by computing similarities between node descriptors (see Section 2.2). As the SemaLink approach builds up a connected, distributed hypermedia, virtual edges have to be computed for all existing pairs of node descriptors world-wide. This, of course, is impossible on the assumption that SemaLink would always access node descriptors on the level of single authors’ hosts.

In the following subsections we describe how a broker/agent architecture [6] overcomes the resource discovery problem of virtual edges, offering users single access points for a whole virtual hypermedia. A combination of resource delivery and resource discovery principles avoids continuous repeated accesses to authors’ home sites though their knowledge takes part in a great variety of retrieval situations.

3.1 Broker Resource Delivery

_Broker_ manage node descriptors and link objects for a certain number of authors. They serve as well-known collection points from which agents (see Section 3.2) obtain their data. A broker is associated to an Internet domain, e.g., a company or a university institute. The main purpose of brokers is to avoid the necessity of agents to access many single hosts in order to obtain data, because this would result in extremely high networking costs. Moreover, brokers should be specialized so that knowledge concerning specific topics can be found at few brokers only.

Authors edit nodes and links at their local hosts. To publish their work, node descriptors and link objects are sent to a suitable broker, i.e., not the documents themselves but only meta-data is copied to brokers. For authors brokers provide the operations _insert_, _update_, and _delete_. By the insert operation new descriptors and links are announced. The update operator may be used to change attribute values of nodes already announced. Node deletion (which should quite seldom) sets a deletion-flag for the corresponding DRL (see Table 1) or LRL (see Table 1). Deleted node descriptors or links are finally removed after a certain period. After each of the described operations the attribute _Date_ is actualized.

Broker clients (e.g., agents) may obtain node descriptors and links by the _retrieve_ operation. As a parameter a select predicate is given. The broker now delivers to the client all descriptors and links that satisfy this predicate. The predicate can be used to address certain authors, institutions, keywords, etc. Especially the attribute _Date_ is used to retrieve only those object descriptors and links that were modified or inserted after the last retrieval operation.

Note that brokers are passive units. They only take action on explicit demand, i.e., they do neither search for data, nor do they select or modify data from authors. Above all they do not compute virtual edges.

3.2 Agent Resource Discovery

In contrast to brokers _agents_ are active units. They connect to brokers and retrieve node descriptors and links, i.e., they search for specific descriptions of new documents and delimited semantic networks. Each agent maintains a list of brokers that are to be visited periodically. By executing the _retrieve_ operation offered by brokers agents generally use the select predicate to retrieve only those node descriptors and links that match their interests. In doing so they, of course, need not fetch all attributes of those nodes. Every agent has a specific profile that is also provided to retrieval clients to facilitate choosing appropriate agents. Normally agents request only those objects that are younger than the last retrieval date. By evaluating the deleted-flag deleted data is recognized and removed from the agent.
Besides node descriptors and links agents also manage virtual edges, i.e., they finally put up the virtual hypermedia. Though similarities could be, at least in principle, computed at navigation time, it makes great sense to compute them only once and materialize them as virtual edge objects at the agents (see Table 1). In this case virtual edge objects are created for every incoming node descriptor retrieved from a broker. Theoretically an agent has to manage up to \( O(n^2) \) virtual edges if it stores \( n \) node descriptors. In practice, however, many similarities are useless for hypermedia navigation as their relevances are less than common values for \( R_{\text{min}} \). This considerably reduces the amount of stored virtual edges.

To enable semantic hypermedia navigation as described in Section 2.2, agents at least should offer the operations select_node, get_links, get_virtual_edges, and get_trans_closure to retrieval clients. The select_node operation allows of accessing node descriptors directly by a declarative queries. Users may, e.g., search for documents from the Intel company that deal with Pentium motherboards and were published after September 95. The get_links and get_virtual_edges operations, on the other hand, are used to directly follow link or virtual edges branching off a given node descriptor.

The operations described so far implement the hypertext metaphor. To optimize deductive path determination the operation get_trans_closure may be used. By this operation clients can fetch subnets of the virtual hypermedia. Starting from a given node descriptor (DRL) an agent returns to the client all paths whose lengths do not exceed a certain value \( L \). Such nets may be the basis for an efficient deduction process at the retrieval client and can be seen as a preselection. Depending on \( u_L \) and \( R_{\text{min}} \) an appropriate value for \( L \) easily can be calculated.

### 3.3 Retrieval Clients and Document Servers

**Retrieval clients** are the user front-end of the SemaLink approach. They are responsible for user-interaction and manage the communication with agents and document servers. Retrieval clients obtain meta-data (i.e., node descriptors, virtual edges, and links) from an agent while they access node contents directly from document servers. **Document servers** are located at the authors’ hosts. Their main task is to deliver node contents to given CRLs. As the whole resource discovery process only involves retrieval client and agent, document servers are only accessed when a user decides to have a document displayed.

Retrieval clients also manage the user model, i.e., they provide the indexing functions \( u_S \), \( u_W \), \( u_L \), the relevance bound \( R_{\text{min}} \) and a set of rules for deductive path determination (see Section 2.3). Nodes may be accessed directly by querying. The retrieval client therefore executes the select_node operation of the agent. Resulting node descriptors coming from the agent are displayed in a ranked list according to their similarity to the query. If the user decides to have a node contents displayed, the retrieval client connects to the corresponding document-server, requests the node contents and displays it.

If the user now looks for related documents, the DRL is transmitted to the agent. To obtain single step navigation, the operations get_links and get_virtual_edges of the agent are executed. Links are presented to the user with their type and weight. Activating such a link or virtual edge (which may be regarded to be a link of type similar) yields directly connected node descriptors that are requested again by means of the select_node operation. If paths are computed deductively, the get_trans_closure operation delivers a subnet containing the initial node as fact basis for the deduction. The path determination then is processed locally at the retrieval client. Resulting path relationships are again presented to the user with their computed relevances and types. They may be activated in an analogous manner.
4 Summary and Outlook

A new approach for semantic navigation through large distributed document spaces has been proposed in this paper. We discussed how semantic relationship knowledge and multimedia documents can be combined to a coherent, knowledge-based hypermedia. This hypermedia can be navigated by computed relationships that take into consideration user-specific rules and preferences. A broker/agent architecture solves resource discovery problems arising with the introduction of virtual edges that express similarities between objects of the hypermedia.

Although the basic approach is already suitable for a prototype implementation, some issues can be further refined. Currently we are developing an enhanced path deduction algorithm that considers subsumption effects and passes local attribute information along paths. Furthermore composite documents are essential for practical usage of the SemaLink approach. Hypertext anchoring on a more fine-grained level has to be integrated. To further improve resource discovery inter-agent-communication could be used to compute virtual edges across agent boundaries.

5 References