Association Index Architecture for Information Brokers

B.C.M. Wondergem, P. van Bommel, and Th.P. van der Weide
Computing Science Institute, University of Nijmegen
Toernooiveld 1, NL-6525 ED, Nijmegen, The Netherlands
tel: +31 24 3653147, fax: +31 24 3553450
E-mail: bernd@cs.kun.nl

Technical Report CSI-R9820

Keywords: Information Retrieval, Information Filtering, Intelligent Agents, Information Brokers, Information Discovery, Knowledge Structure.

Abstract

Information Discovery (ID) is the synthesis of Information Retrieval (IR) and Information Filtering (IF). In ID, broker agents act as intermediaries between user agents and source agents. Information about user interests and documents in sources can be modeled by 2-level hypermedia representations. These representations allow navigational mechanisms which have proven their effectiveness in IR applications.

Broker agents should thus combine two 2-level hypermedia representations to obtain an overall information structure necessary for the synthesis of IR and IF. For this, we propose the so called *Association Index Architecture* (AIA) which consists of two 2-level hypermedia representations which are connected through a third level which is coined the *association index*. The AIA thus forms a 3-level hypermedia representation. Broker agents can perform actions in the AIA to implement their IR and IF related tasks. The AIA is shown to be a general symbolic architecture for combining knowledge by illustrating how a number of ID applications can be performed in it.

Contents

1 Introduction

The quest for relevant information has given rise to two major plans of attack: Information Retrieval (IR) (see [Rij75]) and Information Filtering (IF). In IR, users formulate their information need in a query which is fired off to the IR system that, after processing it, returns the set of correponding relevant documents. In IF, newly created or altered documents are presented to the filtering system which forwards them to interested users. IR and IF, which are elaborately compared in [BC92], have received great attention over the past decades.

Information Discovery (ID) (see [WBHW97]) is the synthesis of IR and IF. In ID, a networked population of users having dynamic information needs is considered. The information needs are to be satisfied with documents out of dynamic information sources. To facilitate this process, information brokers act as intermediaries between users and sources. Information brokers compare the wishes of users with the available information in the sources. In addition, they are able to cope with dynamics from both users and sources, provide fairness (concerning e.g. privacy) in a biased world as a trusted third party, and flexibly cope with different kinds of sources and users. Figure 1 sketches the ID paradigm at a conceptual level.

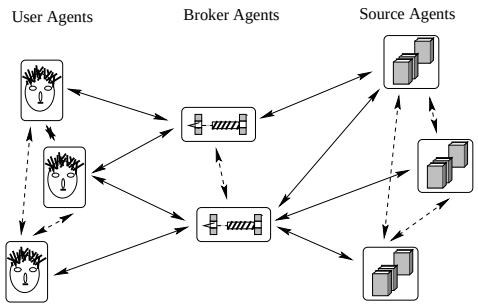

Figure 1: Information Discovery with Agents

Since agent technology (see e.g. [WJ95]) suits the ID paradigm perfectly (especially communication, reactiveness, and proactiveness - see [WBHW98]), the entities of the ID paradigm in figure 1 are modeled as agents: user agents, broker agents, and source agents, respectively. The community of agents in ID forms a multi-agent system. The arrows in figure 1 represent communication. Note that information brokers are true *middle-agents* (see e.g. [DS97, KH97]) since they are intended as a communication facility between users and sources. User agents within ID are focussed on in [Sim97], whereas natural language processing aspects of documents in source agents are elaborated on in [AWBK97]. It is not the goal of this article to analyse the interaction and communication between agents. Rather, we focus on the knowledge that agents need to interact.

In this article, we focus on broker agents, also called information brokers. As information brokers act as middle-agents, they deal with both users and sources. The sources of several recent IR systems are modeled as *2-level hypermedia*

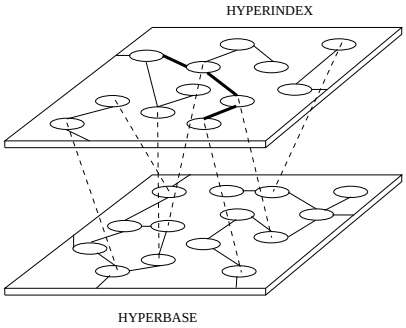

Figure 2: 2-Level Hypermedia Representation.

([BW90, ACG91]) consisting of two levels or layers (see figure 2). The set of documents to be queried constitutes the base layer. On top of that, an overview structure called a hyperindex is created based on the contents of the documents. This has proven successful in e.g. supporting navigational query formulation mechanisms such as Query by Navigation (QBN) (see [Bru93]) and berrypicking (see [Bat89]). For IF, a similar approach can be taken by modelling users and their interests as a 2-level hypermedia. The users form the base layer on top of which a filter topology serves as hyperindex. The filter topology constitutes hierarchically composed user profiles in which documents navigate to interested users in the user base layer (see e.g. [RIS+94]).

In order for broker agents to be effective intermediaries, they should have knowledge about both users and sources. That is, their knowledge should comprise the combination of both user and source 2-level hypermedia representations. However, an architecture for the combination of two 2-level hypermedia representations has not been developed yet. This combination architecture, incorporating both user interests and document contents, would provide a standardised and uniform framework in which the knowledge of broker agents can be structurally processed. Next, basic actions in this combination framework would define (discrete transitions in) the state of the knowledge of broker agents. Since the capabilities of broker agents can be defined in terms of these basic actions, this also lies the basis for inter-broker cooperation. Thus, the combination architecture defines the basis for collaborative IR and IF. Furthermore, navigational mechanisms known in IR, which exploit the 2-level hypermedia representation, can then be adopted for middle agents. This enables long-term practical experience with search strategies to be fluently incorporated in broker agents. Finally, ID concepts as automatic query expansion, profile adaptation, and query generation can then be integrated in the combination architecture by describing them as series of basic actions upon the combined information. This also allows broker agents to perform autonomous or proactive IR. Concluding, a combination architecture for user interests and document content is needed to support both techniques from ID as well as agent related tasks within ID.

This resulting knowledge representation architecture should satisfy a number

of criteria. First, it should combine user and source knowledge in a way that preserves accessibility of the individual hypermedia representations without loss of information. In addition, it should support internal broker actions for dealing with processing this knowledge. Finally, it should be powerful enough to deal with a variety of methods and techniques from IR and IF.

To fill the absence of a suitable combination architecture for two 2-level hypermedia representations, we propose the so called *Association Index Architecture* (AIA) as knowledge representation mechanism for broker agents. The AIA is a 3-level hypermedia, consisting of two 2-level hypermedias for users and sources and an additional layer called the association layer. This layer, which consists of a so called *association index*, forms the connection between the two individual 2-level hypermedias by describing their maximal overlap.

Although the approach taken in this article is based on the structured language of index expressions, it is applicable for every descriptor language which can be equipped with a subexpression relation. Furthermore, it can easily be generalised to combine any number of 2-level hypermedia representations.

This paper is organised as follows. Section 2 formulates the basic building blocks used to denote user interests and document content: index expressions and lithoids. In section 3, the association index is defined in terms of lithoids describing user interests and document contents. The association index forms the third layer of the AIA. Section 4 elaborates on actions broker agents can perform within the AIA. In section 5, a number of applications of the AIA in ID are provided. Finally, section 6 provides conclusions and directions for further research.

2 2-Level Hypermedia Representations

This section provides background on 2-level hypermedia representations which are constructed from *index expressions*. Index expressions are used to describe user interests and document contents. The choice for index expressions is motivated as follows. Noun phrases are seen as a basic unit of human thought (see [ATKW98, Win83]). The structure of such phrases is rather complex. Index expressions are a reasonable approximation of noun phrases, and therefore used to represent user interests and document contents. From index expressions, navigational overview structures called lithoids are constructed, as described below.

2.1 Descriptor Language: Index Expressions

The structured characterisation language of *index expressions* is defined below. Index expressions are based on *terms*, such as keywords, concept names, and denotations of attribute values, and on *connectors*, representing relations between terms in the form of prepositions and gerunds.

Definition 2.1

Index Expression Language *Let T be a set of terms and C be a set*

of connectors. The language $\mathcal{L}_{(T,C)}$ of index expressions over T and C is defined by the following syntax as given in [Bru93]:

IExpr	$\rightarrow$	ϵ | NExpr
NExpr	$\rightarrow$	Term {Connector (NExpr)}*
Term	$\rightarrow$	t, t $\in T$
Connector	$\rightarrow$	c, c $\in C$

□

If no confusion about T and C exists, we will write $\mathcal{L}$ for short. Example index expressions are ϵ, wind ∘ surfing, and surfing in Australia. The so-called null-connector is denoted by ∘ and is used to describe multi-word nouns (collocations) such as hot ∘ dog and health ∘ care. The empty index expression, denoted by ϵ, is an example of a universal descriptor ([Ber98]) since it can represent an unspecified information need.

Index expressions can be broken down into subexpressions. An index expression j is a subexpression of another index expression i, denoted $j \preccurlyeq i$, iff j is contained in i. In other words, the language of index expressions $\mathcal{L}$ is equipped with a subexpression relation $\preccurlyeq \subseteq \mathcal{L} \times \mathcal{L}$. This subexpression relation on index expressions is reflexive, antisymmetric, and transitive (see [Bru93]). This, in turn, means that $(\mathcal{L}_{(T,C)}, \preccurlyeq)$ is a poset. In addition, the strict (irreflexive) variant of the subexpression relation is denoted by $\prec$. Finally, direct subexpressions are defined as index expressions that directly precede one another. Formally, the direct subexpression relation $\prec_d \subseteq \mathcal{L}_{(T,C)} \times \mathcal{L}_{(T,C)}$ is defined as: $j \prec_d i \Leftrightarrow j \prec i \,\mathtt{and}\, \neg\exists_{k \in \mathcal{L}} [j \prec k \,\mathtt{and}\, k \prec i]$. The language of index expressions, the mentioned subexpression relations, and a number of their properties are formalised and proven in [WBW98b].

As running example, we consider index expressions about windsurfing, Australia, and surfing the Internet. For instance, surfing is a subexpression of surfing in Australia and surfing in Australia is a subexpression of wind ∘ surfing in Australia. Moreover, surfing also is a subexpression of wind ∘ surfing in Australia. It is no direct subexpression, however, since the difference is more than a single term. In addition, wind is not a subexpression of surfing in Australia, nor is wind in Australia a subexpression of wind ∘ surfing in Australia. The latter case is prohibited by the structure of index expressions. Note that the empty index expression ϵ is a subexpression of every index expression and a direct subexpression of all terms.

An augmented form of index expressions is given in [Ber98]. This augmented form allows adjectives in index expressions. In addition, normalisations of index expressions are considered. The normalisations are modeled by a canonical form for index expressions and a similarity relation that expresses semantical equivalence. The similarity relation is denoted by $\sim \subseteq \mathcal{L}_{(T,C)} \times \mathcal{L}_{(T,C)}$, where $i \sim j$ means that the meaning of i and j is the same, i.e., they express the same informational content. The normalisations are based on three ideas concerning the information index expressions carry. First, the order in which subexpressions

appear is considered to be irrelevant. Second, it is claimed that the meaning of index expressions does not change when subexpressions are repeated. Finally, adding or deleting the empty index expression from a nonempty index expression is assumed not to change its meaning. The similarity relation $\sim$ is an equivalence relation for index expressions. Augmented index expressions and their normalisations can be directly used in our formalism by exploiting the canocical forms in our constructions.

It should be noted that the approach given here is tailor-made for the structured descriptor language of index expressions. However, the idea is applicable to every *structured descriptor language* ([Won96]). A structured descriptor language is a set of descriptors equipped with a subexpression relation with the properties of a partial order.

2.2 Lithoids as Hyperindices

In this section, the navigation through a set of index expressions (and their subexpressions) is considered. This navigation serves as a basis for Information Discovery. Index expressions enable us to navigate through collections of users and documents via auxiliary access structures such as *hyperindices* (known from e.g. Query by Navigation [BW92]). The base collections of users and documents equipped with a hyperindex leads to a 2-level hypermedia structure (see e.g. [BW90, ACG91]).

Navigation is supported by an auxiliary structure which provides access to the index expressions. Such a structure, called a hyperindex, is a graph-like structure in which navigation between nodes takes place over the edges. A special instantiation of a hyperindex is a *lithoid* ([Bru93, BW90]), which is defined in terms of the underlying language of index expressions $\mathcal{L}_{(T,C)}$ and the subexpression relations for index expressions.

Definition 2.2

Lithoid *Let $I \subseteq \mathcal{L}_{(T,C)}$ be a non-empty set of index expressions. Then, the lithoid for I is a graph $L_I = \langle V, E \rangle$ consisting of vertices V and edges E, where*

- $V =_{\mathtt{def}} \{ j \in \mathcal{L}_{(T,C)} \mid \exists_{i \in I} [j \preccurlyeq i] \}$, *i.e., the smallest set containing all subexpressions of the elements of I, and*
- $E =_{\mathtt{def}} \prec_d \cap\, V \times V$, *i.e., the smallest set containing edges between vertices out of V that represent direct subexpressions.*

In addition, define $L_\varnothing =_{\mathtt{def}} L_{\{\epsilon\}}$. That is $L_\varnothing = \langle \{\epsilon\}, \{(\epsilon, \epsilon)\} \rangle$. □

Note that for every lithoid $L_I = \langle V, E \rangle$, we have $I \subseteq V$ by reflexivity of $\preccurlyeq$.

Figure 3 provides example lithoids. In part (a) of figure 3, the lithoid $L_{\{\mathsf{AUS}, \mathsf{WIND} \circ \mathsf{SURF}\}}$ is given. Part (b) presents lithoid $L_{\{\mathsf{INT} \circ \mathsf{SURF}\ in\ \mathsf{AUS}\}}$.

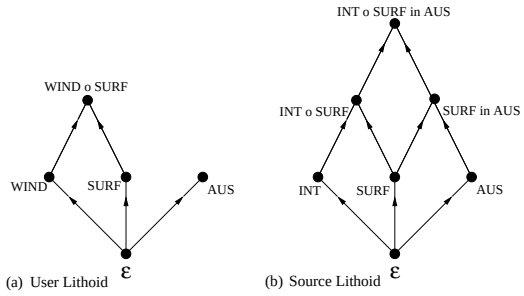

Figure 3: Example Lithoids

2.3 Source of Documents

We consider a set of documents $\mathcal{D}$. The source agents of the ID paradigm can distinguish themselves by differences in known documents and in characterisation techniques. Therefore, source agents are modeled by a *characterisation relation*. The set of source agents is denoted by S. Each source agent $\chi \in S$ is a characterisation relation $\chi \subseteq \mathcal{D} \times \mathcal{L}$. The expression $\cup S$ denotes the *overall characterisation relation*.

The expression $(d, l) \in \chi$ intuitively means that, according to source agent χ, document d is *about* index expression l. For example, $(d, \mathsf{Surfing}\, in\, \mathsf{Australia}) \in \chi$ means that, according to source agent χ, document d is about surfing in Australia. Aboutness, see for instance [Hui96], describes the topic of a document's content.

The following standard notations are used in this article. Given a source agent χ, $d\chi$ denotes the characterisation of document $d \in D$ according to source agent χ. In addition, χi denotes documents that are about descriptor i according to source agent χ. Furthermore, $D\chi$ denotes the summary of document collection D, that is, all descriptors used by χ to characterise documents of $\mathcal{D}$. Also, χI denotes the collection of documents that are about at least one descriptor from $I \subseteq \mathcal{L}$.

Note that $L_{\mathcal{D}(\cup\chi)}$ denotes the lithoid for the complete document set $\mathcal{D}$ and the overall characterisation relation. However, if $\mathcal{D}$ is considered to consist of all documents on the Internet, the sheer size of $\mathcal{D}(\cup\chi)$ and the frequent changes in documents cause serious problems in computing the complete and exact lithoid $L_{\mathcal{D}(\cup\chi)}$. This problem is tackled in ID by introducing a number of cooperating agents, each acquiring part of the lithoid mentioned.

Example 2.1

Figures 4 and 5 provide example 2-level hypermedia representations. The user part, depicted in figure 4, represents user interest relation

$$\eta = \{(u1, \mathsf{Wind} \circ \mathsf{Surf}), (u2, \mathsf{Aus}), (u3, \mathsf{Aus}), (u3, \mathsf{Surf})\}$$

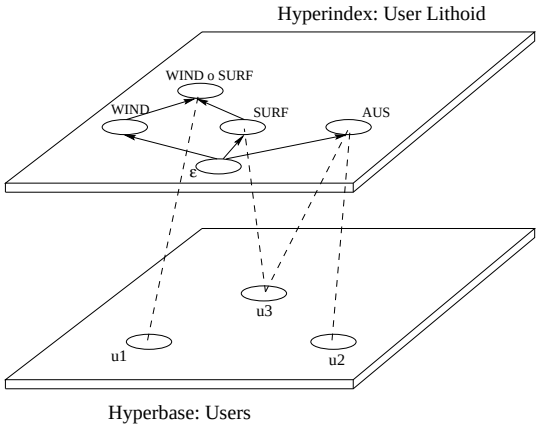

Figure 4: Example 2-level Hypermedia Representation for Users.

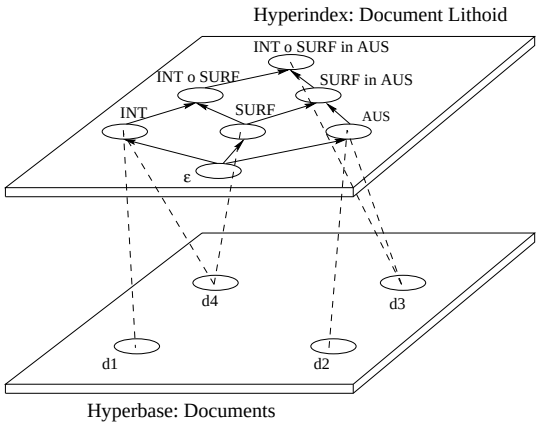

Figure 5: Example 2-level Hypermedia Representation for Documents.

The source part, depicted in figure 5, represents the characterisation relation

$$\chi = \{(d1, \mathsf{Int}), (d2, \mathsf{Aus}), (d3, \mathsf{Aus}), (d3, \mathsf{Int} \circ \mathsf{Surf} in \mathsf{Aus}), (d4, \mathsf{Int}), (d4, \mathsf{Surf})\}$$

The lithoids that are constructed from these sets form the hyperindices.
□

2.4 Population of Users

Without loss of generality, users are modeled in a symmetrical way to sources. This enables us to abstract from the actual derivation of user interests and concentrate on the result of this process. We consider a set of users $\mathcal{U}$ that are in need of information. User agents derive user models, which include a

description of the user's interests. Other tasks of user agents are to submit queries to appropriate brokers, to present the resulting documents to the user, and to ask the user for relevance feedback. In this article, we focus on the derived user interests. Therefore, user agents are modeled by a *user interest relations*. The set of user agents is denoted by U. Each user agent $\eta \in U$ is modeled by a user interest relation: $\eta \subseteq \mathcal{U} \times \mathcal{L}$. The expression $\cup U$ denotes the *overall interest relation*.

The expression $(u, l) \in \eta$ means that, according to user agent η, user u is interested in the topic described by index expression l. That is, that user u's information need is reflected in l. For instance, $(u, \mathsf{Surfing}\mathit{in}\mathsf{Australia}) \in \eta$ means that, according to user agent η, user u is interested in documents about surfing in Australia. The derivation of user interests is dealt with in e.g. [JSSW95, Ric79].

Notational shorthands are introduced similarly to source agents. Given a user agent η, $u\eta$ denotes user agent η's profile of user $u \in \mathcal{U}$. In addition, ηi denotes the set of users interested in descriptor i according to η. Furthermore, $U\eta$ denotes the group profile of all users in U, that is, the set of descriptors agent η uses to profile the users from U. Also, ηI denotes the interest group of I: all users that are interested in a descriptor from I according to η.

Note that $L_{\mathcal{U}(\cup U)}$ denotes the lithoid for the complete user set $\mathcal{U}$ and the overall profiling relation. However, the number of users attached to an ID system and the changing nature of their information needs cause serious problems in computing and maintaining this complete and exact lithoid. In ID, multiple user agents each derive the interests of a limited number of users. By cooperating, the complete lithoid is approximated.

2.5 Knowledge of Broker Agents

The knowledge of broker agents considers users and their information needs and documents and their contents. As mentioned in the previous sections, both user interests and document contents are described by index expressions. By coupling these index expressions to the users or documents they describe, 2-level hypermedia representations are obtained. Broker agents thus have two 2-level hypermedia representations at their disposal: one for users (see figure 4(a)) and one for documents (see figure 4(b)).

We consider a set of broker agents B. Each broker agent $b \in B$ has knowledge about users, modeled by a user interest relation η_b, and about documents, modeled by a characterisation relation χ_b.

We define two auxiliary functions delivering the sets of index expressions that constitute the broker's knowledge of users and of documents. This information forms the basis of the hyperindex layer. In other words, the user and source lithoids are constructed from it.

User interests:	$\mu(b) =_{\mathtt{def}} \pi_{\mathcal{L}} \eta_b$
Document contents:	$\delta(b) =_{\mathtt{def}} \pi_{\mathcal{L}} \chi_b$

In addition, two similar functions define the known users and documents. The entities rendered by these functions reside in the base layers of the two

2-level hypermedia representations.

$$\begin{array}{ll} \text{Known users:} & \overline{\mu}(b) =_{\texttt{def}} \pi_{\mathcal{U}}\eta_b \\ \text{Known documents:} & \overline{\delta}(b) =_{\texttt{def}} \pi_{\mathcal{D}}\chi_b \end{array}$$

The knowledge of brokers is also denoted as a tuple $K_b = \langle \eta_b, \chi_b \rangle$. Brokers start without knowledge, i.e., $K_b = \langle \varnothing, \varnothing \rangle$. By processing user queries (IR) and routing documents to interested users (IF) brokers augment their knowledge as described in section 5. Section 4 elaborates on the actual use of this knowledge and describes actions broker agents can perform in it.

We argue our formalisation with the following underlying thoughts. Instead of defining broker agents in terms of user profiling and document characterisation relations that are supported by other agents of the ID paradigm, we introduce new relations η_b and χ_b. In this way, each broker can have his own view on the world and is not restricted to the information available from user and source agents. The knowledge η_b and χ_b of brokers will, of course, be based on information gained from user and source agents, but we do not demand that they commute. In addition, aspects of time (for instance, delayed processing of changes), other sources of information (for example, thesauri and dictionaries), and different ways of matching can be described now each broker agent is allowed to have his own personal view on user interests and documents' contents. Arguments of limited space and time also support a possible discrepancy between broker's knowledge η_b and χ_b and delivered information through user agents' user interest relations η and source agents' document characterisation relations χ. These arguments correspond to viewing agents with bounded rationality (see e.g. [SV97]). This also means that broker agents do not *have* to know all user and source agents.

Possible augmentations to our approach may involve knowledge about other agents. Broker agents can, for example, store the name of the agent that supplied some information together with this information. This opens the possibility of agent selection based on knowledge about their capabilities and requests. For instance, broker agents may then be able to derive which source agents contain many relevant documents about a specific topic. The broker's first guess would then be to contact this source agent first.

By cooperating, agents can combine their individual AIAs and form a distributed global AIA. That is the second perspective from which the AIA can be viewed. That is, as overall structure for all agents in the ID paradigm. It then contains all information that broker agents have about users and documents participating in ID. Ideally, this means that the Association Index Layer is based on user lithoid $L_{P(\mathcal{U})}$ and source lithoid $L_{\chi(\mathcal{D})}$, exactly modeling the information provided by user and source agents. However, the argumentation just above conveys the idea that this will rarely be the case.

3 Association Index Architecture

Broker agents facilitate the information exchange as intermediaries between user agents and source agents. The knowledge of information brokers thus concerns requests (interests) and offers of information. This was described in the previous section. The knowledge of the broker enables the construction of a 3-level hypermedia representation, the *Association Index Architecture*.

This section is organised as follows. The main ideas behind the AIA are sketched in section 3.1. The first two layers of the AIA, (1) the base layer consisting of documents and users and (2) lithoids describing their contents and interests, were described in section 3. This section deals with the third layer: the association layer. The association layer consists of an association index, which is a graph-like structure built from nodes and edges. The nodes are described in section 3.2 and the edges in section 3.3. The association index is then defined in section 3.4. Finally, two related approaches are described in section 3.7.

3.1 Main Ideas and Intuition

Broker agents know of users and documents. They represent their knowledge about this in two 2-level hypermedia representations (see figure 4). To fit this knowledge into a single architecture, the Association Index Architecture is developped. It consists of the two 2-level hypermedia representations augmented with a third layer which forms the connection.

The architecture a broker constructs thus consists of three layers. The lowest layer, the base layer, consists of users and documents. The middle layer, called the lithoid layer, consists of the user and source lithoid. The third and topmost layer consists of the broker's association index. The resulting structure of layers is called the association index architecture and is depicted in figure 6.

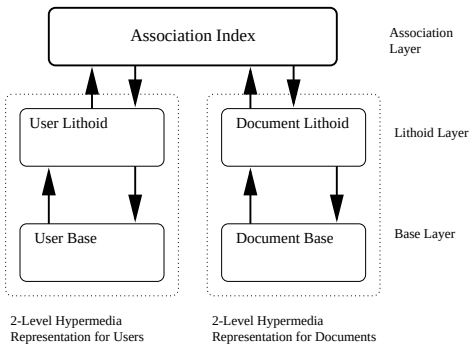

Figure 6: The Association Index Architecture, a 3-Level Hypermedia

The association index architecture of each broker consists of three layers:

Association Index Layer. The topmost layer consists of the association index. The association index forms the connection between the two in-

dividual 2-level hypermedia representations that constitute the lower two layers. The association layer is defined in terms of both hyperindices in the lithoid layer.

Lithoid Layer. The lithoids that are constructed based on the broker agent's user interest relation and characterisation relation comprise this middle layer.

Base Layer. The lowest layer consists of the users and documents that are known by the broker agent.

The arrows in figure 6 depict possible transitions between layers in the AIA. Those are described in section 4.

3.2 Association Nodes

As information brokers essentially are intermediaries between users and documents, they reside in the combination of both worlds. Consequently, a broker agent b combines his information η_b about users and χ_b about documents.

The language in which this combination is described is called the *Association Language*, denoted by $\mathcal{L}_{\mathtt{AI}}$. This language expresses (1) the overlap of user needs and offered documents in so called *core nodes*, (2) unmatched user needs in *user nodes*, and (3) unmatched document's contents in *source nodes*.

Definition 3.1

Language of Association Nodes *Based on the language of index expressions $\mathcal{L}$, the* association language $\mathcal{L}_{\mathtt{AI}} \subseteq \mathcal{L} \times \mathcal{L}$ *is defined as:*

$$\mathcal{L}_{\mathtt{AI}} =_{\mathtt{def}} \bigcup_{l \in \mathcal{L}} \{(l,l),(l,\epsilon),(\epsilon,l)\}$$

Nodes of the form (l,l) are called core nodes, *nodes (l,ϵ) for which $l \neq \epsilon$ are called* user nodes, *and nodes (ϵ,l) for which $l \neq \epsilon$ are called* source nodes. □

The knowledge of the broker defines the nodes that appear in the broker's association index. These nodes are coined the *broker's association nodes*. The broker constructs these nodes by forming core nodes, user nodes, and source nodes as follows.

A broker agent b constructs a core node (l,l) for each index expression l that appears in both his user lithoid $L_{\mu(b)}$ as well as in his source lithoid $L_{\delta(b)}$. He constructs user nodes (l,ϵ) for index expressions l that do appear in the user lithoid but not in the source lithoid. Finally, he constructs source nodes (ϵ,l) for index expressions l that do appear in the source lithoid but not in the user lithoid. This is reflected in the next definition. Note that the three types of nodes are disjoint.

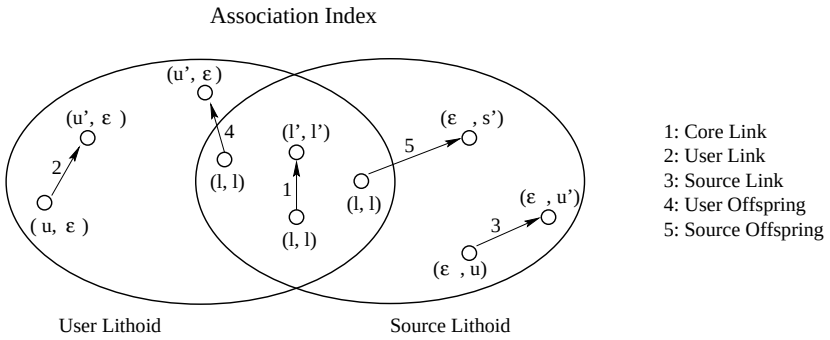

Figure 7: Five Types of Links in Association Index.

Definition 3.2

Broker's Association Nodes *Consider a broker $b \in B$. Based on his user lithoid $L_{\mu(b)} = \langle V_{\mu(b)}, E\rangle$ and his source lithoid $L_{\delta(b)} = \langle V_{\delta(b)}, E'\rangle$ the set of b's* association nodes $V_b \subseteq V_{\mu(b)} \times V_{\delta(b)}$ *is defined as:*

$$\begin{array}{lll} V_b & =_{\mathtt{def}} & \{(l,l) \mid l \in V_{\mu(b)} \cap V_{\delta(b)}\} \quad \textit{(core nodes)} \\ & \cup & \{(l,\epsilon) \mid l \in V_{\mu(b)} - V_{\delta(b)}\} \quad \textit{(user nodes)} \\ & \cup & \{(\epsilon,l) \mid l \in V_{\delta(b)} - V_{\mu(b)}\} \quad \textit{(source nodes)} \end{array}$$

□

The broker's association nodes form the nodes of the association index. The edges that run between association nodes are the topic of the next section.

3.3 Association Edges

The edges in the association index are given by the *association relation* $\prec\!\!\prec_d \subseteq \mathcal{L}_{\mathtt{AI}} \times \mathcal{L}_{\mathtt{AI}}$.

Definition 3.3

Association Relation *The association relation $\prec\!\!\prec_d \subseteq \mathcal{L}_{\mathtt{AI}} \times \mathcal{L}_{\mathtt{AI}}$ is defined by equations (1)..(5) which are given below.* □

There are five types of links, which is illustrated in figure 7. Core links run between two core nodes. User and source links run between two user and source nodes, respectively. Finally, offspring nodes run from a core node to either a user or a source node. The exact definitions are provided below. The numbers of the equations correspond to the cases of figure 7.

The first type are *core links*, which run between two core nodes. Only core nodes whose index expressions are direct subexpressions are connected with a core link:

$$(l,l) \prec\!\!\prec_d (l',l'), \text{ iff } l \prec_d l' \textbf{(core links)} \tag{1}$$

The second type of link, *user links*, run between two user nodes. Note that the ϵ on the right hand place and the other constraints guarantee that both nodes are user nodes, i.e., that their left hand places cannot be empty. The constraint $u \prec_d u'$ makes sure that the destination of a user link is (one step) more complicated than its source.

$$(u, \epsilon) \prec\prec_d (u', \epsilon), \text{ iff } u \prec_d u' \texttt{and} u \neq \epsilon \textbf{(user links)} \tag{2}$$

For the third type, *source links*, similar arguments apply as for user links. This similarity is also found between the last two types of links.

$$(\epsilon, s) \prec\prec_d (\epsilon, s'), \text{ iff } s \prec_d s' \texttt{and} s \neq \epsilon \textbf{(source links)} \tag{3}$$

The fourth type of link, *user offspring links*, run from a core node to a user node. The left hand place u' of the user node is a direct superexpression of the index expressions l of the core node. Note that (l, l) is a core node and that u' cannot be ϵ making (u', ϵ) a user node.

$$(l, l) \prec\prec_d (u', \epsilon), \text{ iff } l \prec_d u' \textbf{(user offspring)} \tag{4}$$

For the last type, *source offspring links*, similar arguments hold as for the fourth type.

$$(l, l) \prec\prec_d (\epsilon, s'), \text{ iff } l \prec_d s' \textbf{(source offspring)} \tag{5}$$

The links are constructed in a way to enable the broker to use the navigational mechanisms known for lithoids since the association relation has the similar properties as thei direct subexpression relation for index expressions. This is proven in section 3.5.

The association relation describes the links that appear in the association index. More specifically, all edges in an association index conform to this relation. This is defined in the next section.

The association relation can, of course, be augmented with, for instance, semantical domain knowledge. This additional knowledge, for example taken from WordNet (see e.g. [MRF+90]), can be used to personalize the association relation. Richer association relations remain a subject for further research.

3.4 Association Index

The topmost layer of the AIA, the association index layer, consists of an association index. This layer is constructed from the broker's association nodes and the association relation. The association relation defines edges between nodes of the association index. The obtained structure strongly resembles lithoids for index expressions.

Definition 3.4

Broker's Association Index *Given a broker $b \in B$, b's* association index*, denoted $\texttt{AI}_b$, is defined as graph (V, E), where*

- $V =_{\texttt{def}} V_b$, *and*
- E *is* $\prec\!\!\prec_d$ *restricted to* $V \times V$, *i.e.,* $E =_{\texttt{def}} \{(n, n') \in V \times V \mid n \prec\!\!\prec_d n'\}$

□

Example 3.1

Consider the user and source lithoid from figure 3. Figure 8 provides the corresponding association index. In this figure, reflexive and transitive links are not depicted. That is, we adopt the same drawing conventions as for lithoids.

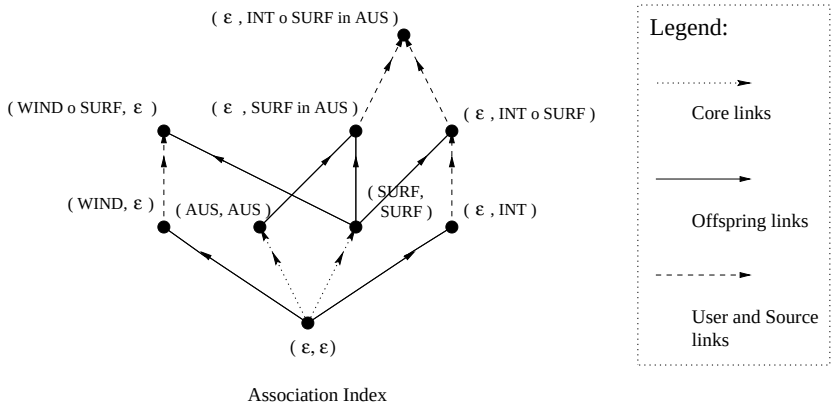

Figure 8: Association Index for Running Example

The lowest element of the association index is the bottom element of the association language: (ϵ, ϵ). *It is smaller, with respect to* $\prec\!\!\prec_d$, *than all other nodes. There are three core nodes,* (ϵ, ϵ), $(\mathsf{AUS}, \mathsf{AUS})$ *and* $(\mathsf{SURF}, \mathsf{SURF})$. *The user nodes are* $(\mathsf{WIND}, \epsilon)$ *and* $(\mathsf{WIND} \circ \mathsf{SURF}, \epsilon)$. *The source nodes are the remaining ones.*

The left side of figure 8 shows the two user nodes connected by a user link. The right side of the figure shows the four source nodes. Core nodes reside in the central part of the figure. We use this as a drawing convention.

Edges are drawn by different arcs in figure 8. Core links, for example, are drawn as thick solid arcs. The two existing core links start at the bottom element (ϵ, ϵ) *and lead to either one of the other core nodes. Offspring links, that depart from a core node and end in a non-core node, are drawn as normal solid arcs. The link from* $(\mathsf{SURF}, \mathsf{SURF})$ *to* $(\mathsf{WIND} \circ \mathsf{SURF}, \epsilon)$ *is an offspring user link. An example of an offspring source link is the edge between* $(\mathsf{AUS}, \mathsf{AUS})$ *and* $(\epsilon, \mathsf{SURF}in\mathsf{AUS})$. *User and source links are drawn as dashed arcs. The link from* $(\mathsf{WIND}, \epsilon)$ *to* $(\mathsf{WIND} \circ \mathsf{SURF}, \epsilon)$ *is a user link. An example source link is the one from* (ϵ, INT) *to* $(\epsilon, \mathsf{INT} \circ \mathsf{SURF})$.

□

Since the lithoids on which the association index is based always contain the empty descriptor, the association index always contains node (ϵ, ϵ) and the according reflexive link. That is to say that $(\{(\epsilon, \epsilon)\}, \{((\epsilon, \epsilon), (\epsilon, \epsilon))\})$ is the smallest association index.

3.5 Properties of the Association Relation

This section provides several properties of the association relation and, thus, of edges in association indices. Our aim was to obtain a navigation structure similar to lithoids of index expressions. In this way, we can build on the knowledge about lithoids and use similar applications as the ones already known for lithoids. To prove we have accomplished a similar structure, we show that the association relation defines is a partial order for the association language, which is also the case for the language of index expressions and the subexpression relation. The proof is given in theorem 3.4 and needs a number of additional theorems.

The first of these tells us that links between certain types of nodes are impossible. For instance, a link from a user node to a source node is impossible.

Theorem 3.1 Impossible Links The following relations between instances of the types of nodes are impossible: (1) user node $\prec\!\!\prec_d$ core node, (2) source node $\prec\!\!\prec_d$ core node, (3) user node $\prec\!\!\prec_d$ source node, and (4) source node $\prec\!\!\prec_d$ user node.

Proof:
We prove the first and third case, the other two find their proof analogously.

- Proof 1: Assume the statement is true, i.e., $(u, s) \prec\!\!\prec_d (u', s')$ and $u \neq \epsilon$ and $s = \epsilon$ and $u' = s'$

  Core Links: $u \prec_d u'$ (from core links) and $u \neq \epsilon$ (starting node is user node) imply that u' consists of at least two keywords and one connector (property of $\prec_d$). By $u' = s'$ (target node is core node), this also holds for s'. This means that $s = \epsilon$ cannot be a direct subexpression of s' (property of $\prec_d$).

  User Links: $u \prec_d u'$ (from user links) means $u' \neq \epsilon$ (property of $\prec_d$). Then, with $u' = s'$ (target node is core node) we have $s' \neq \epsilon$ which contradicts the user link case.

  Source Link: From $u = u' = \epsilon$ (source link) and $u' = s'$ (target node is core node) we obtain $s' = \epsilon$. However, this forbids $s \prec_d s'$.

  User Offspring: $s' = \epsilon$ (user offspring) and $u' = s'$ (target node is core node) imply $u' = \epsilon$ too. This, however, rules out $u \prec_d u'$.

  Source Offspring: $u' = s'$ (target node is core node) and $u' = \epsilon$ (source offspring) imply $s' = \epsilon$ which rules out $s \prec_d s'$.

  Thus, in all cases a contradiction is obtained meaning that the assumption cannot be true.

- Proof 3: Assume the statement is true, i.e., $(u, s) \prec\prec_d (u', s')$ and $s = \epsilon$ and $u' = \epsilon$ and $u \neq \epsilon$ and $s' \neq \epsilon$.

  Core Links: $u' = \epsilon$ (target node is source node) forbids $u \prec_d u'$.

  User Links: $s' \neq \epsilon$ (target node is source node) directly contradicts $s' = \epsilon$ from user link.

  Source Links: $u \neq \epsilon$ (source node is user node) directly contradicts $u = \epsilon$ from source link.

  User Offspring: $u \neq \epsilon$ (source node is user node) and $s = \epsilon$ (source node is user node) contradict $u = s$ from User Offspring links.

  Source Offspring: Same argument as for User Offspring.

  Thus, in all cases a contradiction is obtained meaning the assumption cannot be true.

The possibility of the other types of links is illustrated in figure 8. This figure contains the following links: core-core, core-user, core-source, user-user, and source-source. These links are complementary to the ones mentioned in theorem 3.1, i.e., the cases not mentioned in this theorem are possible.

The next theorem states that when the transfer is made from core nodes to a different type of node, i.e., either user or source nodes, this involves user and source offspring links.

Theorem 3.2 Offspring Links Suppose $(u, s) \prec\prec_d (u', s')$ and let (u, s) be a core node. If (u', s') is a user node, the only applicable part of the definition of the association relation is the case for user offspring links (case 2). If (u', s') is a source node, the only applicable part of the definition of the association relation is the case for source offspring links (case 3).

Proof:
We only prove the first assertion since the second one follows an analogous path of reasoning. Since (u, s) is a core node, we have $u = s$. In addition, since (u', s') is a user node we have $u' \neq \epsilon$ and $s' = \epsilon$.

Core links are impossible, since $s \prec_d s'$ is made impossible by $s' = \epsilon$ (target node is user node).

User links are impossible since $s = \epsilon$ (user link) and $u = s$ (source node is core node) imply $u = \epsilon$ which contradicts $u \neq \epsilon$ from user link case.

Source links are impossible since $u' \neq \epsilon$ (target node is user node) directly contradicts $u' = \epsilon$ (source link).

User Offspring is possible. As an example, consider $u = \epsilon$, $s = \epsilon$, $u' = \mathsf{Surf}$, and $u' = \epsilon$.

Source Offspring is impossible since $u' \neq \epsilon$ (target node is user node) contradicts $u' = \epsilon$ (source offspring).

The next theorem makes a similar statement as the previous one; it also concerns the application of a certain part of the definition of the association relation. This theorem states that links between nodes of the same type are the result of applying case 1 of definition 3.3.

Theorem 3.3 Same Type Links Assume $(u,s) \prec\prec_d (u',s')$ and (u,s) and (u',s') are of the same type. If both nodes are core nodes, then only a core link is possible. Similar statements hold for user nodes and source nodes.

Proof:

1. Two core nodes: $u = s$ and $u' = s'$

   Core Links are possible. Consider, for example, $u = s = \epsilon$ and $u' = s' = \mathsf{Surf}$.

   User Link is impossible since $u \neq \epsilon$ and $u = s$ imply $s \neq \epsilon$ contradicting $s = \epsilon$.

   Source Link is impossible since $s \neq \epsilon$ and $u = s$ imply $u \neq \epsilon$ contradicting $u = \epsilon$.

   User Offspring is impossible since $u' = \epsilon$ and $s' = u'$ imply $s' = \epsilon$ contradicting $s \prec_d s'$.

   Source Offspring is impossible since $s' = \epsilon$ and $s' = u'$ imply $u' = \epsilon$ contradicting $u \prec_d u'$.

2. Two User nodes: $u \neq \epsilon$, $u' \neq \epsilon$, and $s = s' = \epsilon$

   Core Link is impossible since $s = s'$ contradicts $s \prec_d s'$.

   User Links are possible. Consider, for example, $u = \mathsf{Surf}$, $u' = \mathsf{Wind} \circ \mathsf{Surf}$, and $s = s' = \epsilon$.

   Source Link is impossible since $s = s'$ contradicts $s \prec_d s'$.

   User Offspring is impossible since $s = \epsilon$ and $s = u$ imply $u = \epsilon$ which contradicts $u \neq \epsilon$.

   Source Offspring is impossible since $s = s'$ contradicts $s \prec_d s'$.

3. Two Source nodes: This proof follows a similar line of argument as for two user nodes.

The five cases from the definition of the association relation have now been proven to be mutually exclusive (Theorems 3.2 and 3.3). That is, if there exists a link between two nodes in the association index, this link corresponds to exactly one of the five cases of definition 3.3. We now have a clear view on the possible links and from which cases of definition 3.3 they are born. Figure 9 provides a schematic overview of this view.

Definition 3.5

Full Association Relation *Given the association relation $\prec\prec_d$, we define the* full association relation*, denoted by $\prec\prec$, as the transitive and reflexive closure of $\prec\prec_d$. That is,*

$$\prec\prec =_{\mathtt{def}} \prec\prec_d^* \cup \{(I,I) | I \in \mathcal{L}_{\mathtt{AI}}\}$$

where $\prec\prec_d^*$ denotes the transitive closure of $\prec\prec_d$. □

Links		to: User	Core	Source
	User	User Link	–	–
from:	**Core**	Offspring User Link	Core Link	Offspring Source Link
	Source	–	–	Source Link

Figure 9: Overview of Possible Links and Corresponding Cases from Definition 3.3. An – denotes that the mentioned combination is not possible.

The full association relation defines a partial ordering on the association language. For the language of index expressions, the subexpression relation does the same. In this respect, we have thus obtained similar structures.

Theorem 3.4 $(\mathcal{L}_{\mathtt{AI}}, \prec\!\!\prec)$ **is a poset**

Proof:
To prove this, we show that the full association relation $\prec\!\!\prec$ is reflexive, antisymmetric, and transitive. Reflexiveness and transitivity follow trivially from definition 3.5. Antisymmetry is proven below. Suppose $(u, s) \prec\!\!\prec (u', s')$ and $(u', s') \prec\!\!\prec (u, s)$. All five types of links lead to a contradiction, since $\prec_d$ is antisymmetric. For example, in the case of user links we have $u \prec_d u'$ and $u' \prec_d u$ which is contradictory.

3.6 Bounds on the Size of Association Indices

It is proven below that the complexity of creating an entire association index is at least the number of nodes in the association index. The bounds above imply that, for most core sets of index expressions, real-time computation of the association index by broker agents is too time consuming. Therefore, broker agents will only compute the necessary parts, but will virtually have access to the whole association index. This has no influence on the concept of our ideas; it rather sketches its possible implementation.

Theorem 3.5 Bounds on the Size of Association Index Consider a broker $b \in B$ and his user lithoid $L_{\mu(b)}$ and source lithoid $L_{\delta(b)}$. Let $m = \mathtt{max}(|\mu(b)|, |\delta(b)|)$ be the size of the largest set of the broker's user or source knowledge. Furthermore, the maximal number of terms per index expression in $\mu(b) \cup \delta(b)$ is denoted by x. If $|\mathtt{AI}_b|$ denotes the number of nodes in association index $\mathtt{AI}_b$, then $|\mathtt{AI}_b| = O(m2^x)$ and $|\mathtt{AI}_b| = \Omega(m^2)$.

Proof:
We first provide the upper bound of b's association index $\mathtt{AI}_b$. Using [Bru93] we obtain an upper bound on $L_{\mu(b)}$ and $L_{\delta(b)}$: $m2^{x-1}+mx-2m+1$. The maximum size of $\mathtt{AI}_b$ is obtained if $L_{\mu(b)}$ and $L_{\delta(b)}$ only overlap in (ϵ, ϵ).

$$
\begin{array}{lcll}
|\mathtt{AI}_b| & \leq & |L_{\mu(b)}| + |L_{\delta(b)}| - 1 & \text{(no overlap)} \\
 & \leq & (m2^{x-1} + mx - 2m + 1) + (m2^{x-1} + mx - 2m + 1) - 1 & \text{([Bru93] modified)} \\
 & = & m2^x + 2mx - 4m + 1 & \\
\text{thus} & & & \\
|\mathtt{AI}_b| & = & O(m2^x) &
\end{array}
$$

The lower bound for $\mathtt{AI}_b$ is obtained when $\mu(b) = \delta(b)$, implying that the number of vertices in $\mathtt{AI}_b$ will be the same as in either $L_{\mu(b)}$ or $L_{\delta(b)}$, and when all index expressions in $\mu(b)$ are one and the same path expression. From [Bru93] we have $\frac{m(m+1)}{2}$ as lower bound for index expression lithoids.

$$
\begin{array}{lcll}
|\mathtt{AI}_b| & \geq & |L_{\mu(b)}| & (L_{\mu(b)} = L_{\delta(b)}) \\
 & = & |\wp(l)| \text{ for any } l \in \mu(b) & \text{(equality of all } l\text{'s in } \mu(b)) \\
 & = & \frac{m(m+1)}{2} & \text{([Bru93])} \\
\text{thus} & & & \\
|\mathtt{AI}_b| & = & \Omega(m^2) &
\end{array}
$$

It turns out that the computation time of an association index is $\mathcal{O}(xn)$, where n is the number of vertices of the association index and x is the maximal number of terms in the index expressions of $\mu(b) \cup \delta(b)$. One might have expected it to be more costly than that. However, the nice structure of index expressions and the mere three restricted forms of the elements in the association language provide the opportunity to order the generation of the nodes in such a way that creating the links depends, besides on the number of nodes, only on the maximal number of subexpressions that can be created for the index expressions.

Figure 10 depicts the results of an experiment on the growth of the association index for the Cranfield collection. This collection consists of almost 1200 titles. Two lithoids were constructed from it: Lithoid1 from all even titles and Lithoid2 from all odd titles. The lines with these names show the growth of the two lithoids. The line denoted by Common depicts the growth of the number of core nodes. The line denoted by Total gives the total number of nodes in the constructed association index. Finally, the line Worst gives the maximal number of nodes as computed by the upper bound from theorem 3.5.

3.7 Related Work

This section compares the construction of the association index to two related approaches: association graphs and Galois lattices. In these approaches, graph-like structures are constructed as well with nodes that consist of a left hand and a right hand side.

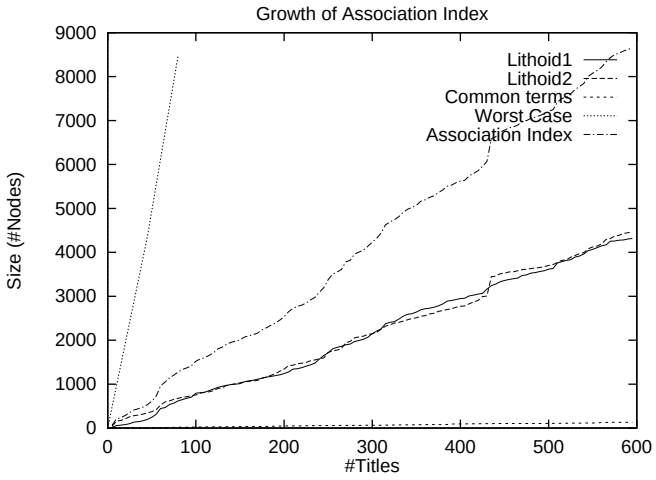

Figure 10: Growth of Association Index for Cranfield Collection.

3.7.1 Association Graphs

The association language expresses the nodes that can appear in an association index. The nodes are more structured than those defined in [MLL92] for association graphs, which makes our association language a special case of the general node language for association graphs. A more enriched association language could be defined, for instance including the complete Cartesian product of the user and source lithoids. However, for our purpose, as sketched by the criteria in the introduction, the defined association language is a fair approach.

The association relation is a special case of the set of edges defined for association graphs in [MLL92]. This is a natural consequence of the fact that the association language is a special case of the association nodes for association graphs.

3.7.2 Galois Lattices

Johannesson describes the use of Galois lattices for supporting schema integration in [Joh95]. The Galois lattices are used to suggest equivalent concepts of the database schema for selection or approval by the user who integrates the two schemata. Here, two conceptual database schemata form the basis of the Galois lattice, which forms the associative structure. This resembles the use of two lithoids as the basis for association indices.

The nodes in these Galois lattices consist of concepts and their context. This means that the two parts of the nodes are of different types. Only so called complete pairs are included. The links in the lattices are also given by a relation that is a partial order, like the subexpression relation.

4 Broker Actions in the AIA

Information broker agents construct an AIA based on their knowledge about user demands and document offers. The AIA forms a stratified representation structure for this knowledge and, in essence, is the broker's view of the world. The main tasks of broker agents, IR and IF, are implemented as series of actions in the AIA. This is the topic of section 5. The actions a broker can perform within the AIA are *navigation primitives* for navigating within a single layer of the AIA and *transitions* between consecutive layers of the AIA.

This section elaborates on these actions. First, in section 4.1, the focus of a broker agent is introduced as a means to let broker agents focus in on a subset of their total knowledge. Section 4.2 describes the navigation primitives and their properties. Finally, section 4.3 does the same for transitions between layers.

4.1 Focus of a Broker

When performing a specific task, broker agents will generally use only a part of their total knowledge. For instance, when processing a user query, the broker may decide not to use knowledge about other users. To enable a broker to select and use the required knowledge only, the *focus* of a broker is introduced. The focus of a broker basically is a subset of his total knowledge.

When navigating or browsing in a 2-level hypermedia, one can reside either in the hyperbase (base layer) or in the hyperindex (lithoid layer), but not in both layers at the same time. A similar argument holds for brokers in the AIA: at any moment, they can reside in only one layer. In addition, in the lower two layers, they can either be in the user part or in the source part. For instance, in the lithoid layer, they can reside in either the user or source lithoid. Thus, the focus of a broker is part of the AIA layer he currently resides in. This is reflected in the next definition.

Definition 4.1

Broker's Focus *For a broker agent $b \in B$, its* focus *in a layer expressed in language L is a tuple $F_{L,b} = \langle V, E \rangle$, where $V \subseteq L$ and $E \subseteq L \times L$, such that*

1. *in the* **Association Layer**, *$\mathcal{L}_{\mathtt{AI}}$ is the layer language, the nodes are restricted by the broker's association nodes: $V \subseteq V_b$, and the edges are restricted to the links of the association index: $E \subseteq \prec\!\!\prec_d \cap V^2$,*
2. *in the* **Lithoid Layer**, *$\mathcal{L}$ is the layer language, we have either $V \subseteq V_{\mu(b)}$ or $V \subseteq V_{\delta(b)}$, and the links are correct lithoid edges: $E \subseteq \prec_d \cap V^2$,*
3. *in the* **Base Layer**, *$U \cup D$ is the layer language, for the nodes we have either $V \subseteq \overline{\mu}(b)$ or $V \subseteq \overline{\delta}(b)$, and there are no links in the base layer: $E = \varnothing$.*

□

Since we focus on the association index, we do not consider edges between users or between documents. Therefore, $E = \varnothing$ in the case of the baselayer. Only a simple augmentation is needed, however, to include these links.

Note that foci can contain edges whose source and target nodes do not belong to the focus. We adhere to this since it allows more nuances of navigation strategies to be captured.

In the next sections, we show how actions change the broker's focus.

4.2 Navigation Actions

Navigation actions are performed within a single layer and effectuate navigation in that layer. Numerous strategies for navigating the layers of a hypermedia system exist. Among these are, for instance, breadth-first, depth-first, and random navigation strategies. We provide basic navigation primitives allowing a wide range of different navigation strategies to be described in our framework.

4.2.1 Navigation Primitives

To cater for elementary changes in the broker's focus, we provide add and delete actions as navigation primitives. We distinguish between adding or deleting a node and doing the same for a link.

Consider a focus $F = \langle V, E \rangle$. An `NAdd`-action augments the current focus F with node n. An `NDel`-action restricts the current focus by removing the designated node. The concrete definitions are given in figure 11.

Node	Link
$\texttt{NAdd}(\langle V, E\rangle, n) = \langle V \cup \{n\}, E\rangle$	$\texttt{LAdd}(\langle V, E\rangle, l) = \langle V, E \cup \{l\}\rangle$
$\texttt{NDel}(\langle V, E\rangle, n) = \langle V - \{n\}, E\rangle$	$\texttt{LDel}(\langle V, E\rangle, l) = \langle V, E - \{l\}\rangle$

Figure 11: Navigation Primitives

Next we consider basic properties of the navigation primitives. An overview is presented in figure 12. Note that these properties are obviously a direct consequence of the previous definitions. They express that adding does not delete and that deleting does not add.

Node	Link
$V \subseteq \texttt{NAdd}(V, n)$	$E \subseteq \texttt{LAdd}(E, l)$
$\texttt{NDel}(V, n) \subseteq V$	$\texttt{LDel}(E, l) \subseteq E$

Figure 12: Properties of Navigation Primitives

4.2.2 Example Navigation Strategies: Query by Navigation

Query by Navigation (QBN) (see [Bru93] and [BW92]) is a navigational mechanism for query formulation. QBN in lithoids usually starts at the bottom of the lithoid and proceeds through (a) refinements, going in the directions of edges, and (b) enlargements, going down within the lithoid. In each step, navigation proceeds at the destination of the last link followed.

A number of QBN variants can be identified with respect to strictness and the starting point (see figure 13). Strictness means that the current point of QBN can be derived from the focus. The restriction that the starting point of QBN should be the bottom element can also be relieved.

The four variants of QBN can be motivated from the point of view of query formulation, i.e., explicitly formulating an information need. For an initially unspecified information need, QBN with bottom start suits best. If the information need is already partly specified, a non-empty starting point saves the user work. The strict form of QBN with bottom start should be seen as functioning like a stack. Refining means putting a link on the stack, whereas enlarging can take place with the topmost link only. The enlargement is thus only valid if the selected link exists in the focus. The strict form with arbitrary start has similar constraints. Refining from a smaller focus than the starting point, i.e. going back up, can only take place by following the downward path in the opposite direction. The same holds for enlargements. Strict forms of QBN need no additional history for inferring the current focus.

QBN	bottom start		
strict	`Refine`(F,l)	$=$	`LAdd`(F,l)
	`Enlarge`(F,l)	$=$	`LDel`(F,l)
non-strict	`Refine`(F,l)	$=$	`LAdd`(F,l)
	`Enlarge`(F,l)	$=$	`LAdd`(F,l)
QBN	arbitrary start		
strict	`Refine`(F,l)	$=$	if $l \in F$ then `LDel`(F,l) else `LAdd`(F,l)
	`Enlarge`(F,l)	$=$	`Refine`(F,l)
non-strict	`Refine`(F,l)	$=$	`LAdd`(F,l)
	`Enlarge`(F,l)	$=$	`LAdd`(F,l)

Figure 13: Four variants of QBN

In this way, QBN fits both in the lithoid layer as well as in the association layer and is expressed by the two navigation primitives. Note that our variants of QBN only concern edges. Including nodes as well enables the formalisation of enriched navigation strategies such as berry picking (see e.g. [Bat89]), where the nodes that are 'picked' can be included as single nodes (see figure 14a).

Finally, we consider the relaxation of the condition for proceeding with navigation, leading to an extended form of QBN. Extended QBN (EQBN) also

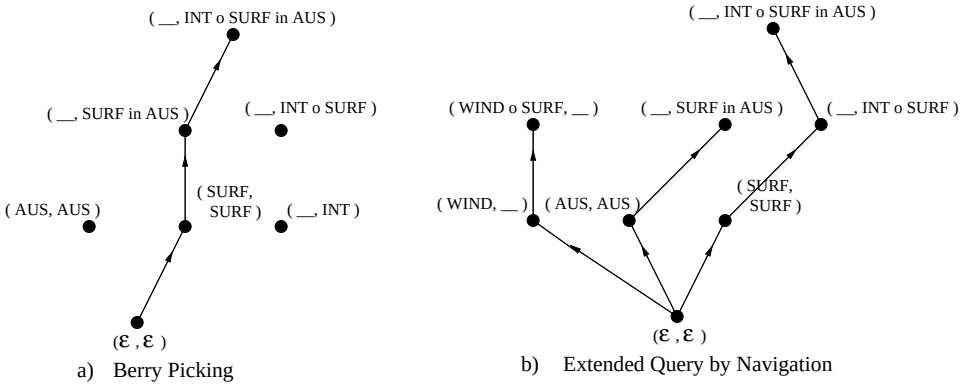

Figure 14: a) Berry Picking, where single nodes are berries picked. b) Extended Query by Navigation.

proceeds through refinements and enlargement. However, navigation can proceed from any node in the focus. This implies that a set of paths is obtained in the navigational layer (see figure 14b). The formalization of QBN given in figure 13 can also serve for EQBN. The difference is that different conditions are to be satisfied when choosing a next link.

4.3 Transitions between Layers

Transitions enable broker agents to travel from one AIA layer to another. Transitions take place between two consecutive layers in the AIA. For instance, a broker agent can make a transition from the lithoid layer to the base layer. However, it is impossible to go directly from the base layer to the association layer, or vice versa.

Transitions provide the resulting focus in the destination layer when given the focus in the original layer. The new focus is, of course, related to the old one. Numerous choices remain as how to define the new focus based on the old one. We provide quite straightforward definitions of the nodes that constitute the new focus. The definitions are conform the idea that the new focus should contain as much information as possible, however restricted to nodes that are based on the old focus. This basic strategy may be further parameterized in order to have more advanced node computations.

Transitions between the lower two layers, i.e., the base layer and lithoid layer, are described in the next subsection. The other transitions, i.e., between lithoid layer and the association layer, are described in the subsequent subsection. In both subsections, the definition of transitions is followed by a look at their properties.

4.3.1 Transitions between Lithoid Layer and Base Layer

Transitions between lithoid layer and base layer are defined similarly for user lithoid and base and for document lithoid and base. The explanation below concentrates on documents. The two directions in the transitions are dealt with in turn. First, we describe upward transitions, i.e., from document base to document lithoid layer. Then, we outline downward transitions from lithoid layer to base layer.

The transitions between lithoid layer and base layer deliver a new focus without edges. This stems from the fact that no relations or dependencies have been included in the base layer. The definition of the base layer does, however, leave room for this by, for instance, using meta-information or computing clusters.

The definition of upward transitions from document base layer to lithoid layer is given in the rightmost part of figure 15 under 'Source Transition'. The upward transition is called `Char` since it bears close resemblance to the characterisation of documents. The transition function `Char` has an argument consisting of a document focus. That is, the focus consists of a set of documents V and possibly relations between them in E. As pointed out before, the new focus contains no edges ($E' = \varnothing$). The set of nodes V' of the new focus consists of the set of index expressions that are used by broker agent b to characterise documents in V of the old focus. That is, set V' is the union, for all documents $d \in V$, of the index expressions l that appear related to d (as a tuple (d, l)) in the broker's knowledge χ_b about documents.

Upward User Transition	Upward Source Transition
`Profile` : $F_{\mathcal{U}} \rightarrow F_{\mathcal{L}}$ `Profile`$(\langle V, E\rangle) = \langle V', \varnothing\rangle$, where $V' = \bigcup_{u \in V} u\eta_b$	`Char` : $F_{\mathcal{D}} \rightarrow F_{\mathcal{L}}$ `Char`$(\langle V, E\rangle) = \langle V', \varnothing\rangle$, where $V' = \bigcup_{d \in V} d\chi_b$

Figure 15: Upward Transitions between Lithoid and Base Layer

Downward transitions from lithoid layer to base layer are specified in figure 16. Again, we concentrate on the case of documents, depicted in the rightmost side of the figure. Two ways of making the downward transition from lithoid to base layer are described, stemming from well-known relevance estimation strategies: overlap and embedment. Overlap transitions deliver documents whose characterisations have some overlap with the current focus. Embedment transitions are more restrictive: they only deliver documents whose characterisations are completely embedded in the current focus.

The downward transition based on overlap is called `DocId`. Transition function `DocId` identifies the documents that share an index expression with the current focus. That is, for all index expressions $l \in V$ in the old focus, documents d are returned for which l also appears as a tuple $(d, l) \in \chi_b$ in the broker's knowledge about documents.

Downward User Transitions	Downward Source Transitions
$\texttt{UserId} : F_{\mathcal{L}} \rightarrow F_{\mathcal{U}}$ $\texttt{UserId}(\langle V, E\rangle) = \langle V', \varnothing\rangle$, where $V' = \bigcup_{l \in V} \eta_b l$	$\texttt{DocId} : F_{\mathcal{L}} \rightarrow F_{\mathcal{D}}$ $\texttt{DocId}(\langle V, E\rangle) = \langle V', \varnothing\rangle$, where $V' = \bigcup_{l \in V} \chi_b l$
$\texttt{UserAbout} : F_{\mathcal{L}} \rightarrow F_{\mathcal{U}}$ $\texttt{UserAbout}(\langle V, E\rangle) = \langle V', \varnothing\rangle$, where $V' = \bigcap_{l \in V} \eta_b l$	$\texttt{DocAbout} : F_{\mathcal{L}} \rightarrow F_{\mathcal{D}}$ $\texttt{DocAbout}(\langle V, E\rangle) = \langle V', \varnothing\rangle$, where $V' = \bigcap_{l \in V} \chi_b l$

Figure 16: Downward Transitions between Lithoid and Base Layer

The other downward transition `DocAbout` which is based on embedment, is more selective in picking documents for the resulting focus. Only those documents whose characterisation is completely contained in the old focus are delivered. That is, only those documents d are returned for which all index expressions $l \in V$ of the old focus are also contained in the broker's knowledge about documents, i.e., $(d, l) \in \chi_b$.

4.3.2 Properties of Transitions between Lithoid and Base Layer

This section touches on properties of the transition functions between lithoid and base layer. First, we consider properties of the overlap and embedment transition functions `DocId` and `DocAbout`. About transitions are more restrictive than identification transitions (see figure 17). Documents that are returned by `DocAbout` are always returned by `DocId` as well, due to the fact that embedment implies overlap. This enables broker agents to use different strategies for finding the interested users in some topic or the documents relevant to that topic.

$$\texttt{DocAbout}(F) \subseteq \texttt{DocId}(F)$$
$$\texttt{UserAbout}(F) \subseteq \texttt{UserId}(F)$$

Figure 17: Properties of Downward Transitions from Lithoid layer to Base layer.

We now consider series of consecutive transitions between the lithoid and base layer. A foreseen use of this lies in query expansion (which is described in more detail in the next section). Consider an index expression $l \in \mathcal{L}$. Broker agents can be interested in other "concepts" (index expressions) that often coexists with l. In processing user queries, for instance, a package of related documents can be sent along as a form of advertising. By following the links downward from l to the user base and, from the new focus, upward again, the broker obtains the complete set of index expressions in which the users interested in l are also interested. This process never results in a smaller focus (see figure 18). Therefore, brokers should be careful not to expand the initial focus

too much by making too many consecutive transitions, since this will lead to too many irrelevant concepts.

$I \subseteq \texttt{Char}(\texttt{DocId}(I))$	$I \subseteq \texttt{Profile}(\texttt{UserId}(I))$
$D \subseteq \texttt{DocId}(\texttt{Char}(D))$	$U \subseteq \texttt{UserId}(\texttt{Profile}(U))$

Figure 18: Broadening focus in consecutive transitions between lithoid and base layer.

Equations similar to those of figure 18 hold for About transitions. Series of consecutive upward and downward transitions between lithoid and base layer expand the focus over and over. Ultimately though, this process converges to the total of broker's knowledge about users or documents.

4.3.3 Transitions between Association Index Layer and Lithoid Layer

The definitions of transitions between the association layer and the lithoid layer are more complicated than the other transitions. This is due to the dependence between the languages of foci in both layers. The languages of foci for the lithoid layer and base layer are not dependent. The edges in the lithoid layer are defined in terms of those in the association layer and vice versa. In defining the set of edges of the new focus, the edges of the old focus play an important role (see figures 19 and 20).

Again, the cases for users and documents are symmetric. This time, we concentrate on the case for users. First, the downward transitions are described, followed by those upward. Intuition is sketched first, followed by definitions and their explanation.

Upward Transitions
$\texttt{UBeamUp} : F_{\mathcal{L}} \mapsto F_{\mathcal{L}_{\texttt{AI}}}$ $\texttt{UBeamUp}(\langle V, E\rangle) = \langle V', E'\rangle,$ where $V' = V_b \cap \bigcup_{u \in V} \{(u,u),(u,\epsilon)\}$ $E' = \{((u,s),(u',s')) \in (\prec\!\prec_d \cap V' \times V') \mid (u,u') \in E\}$
$\texttt{SBeamUp} : F_{\mathcal{L}} \mapsto F_{\mathcal{L}_{\texttt{AI}}}$ $\texttt{SBeamUp}(\langle V, E\rangle) = \langle V', E'\rangle,$ where $V' = V_b \cap \bigcup_{s \in V} \{(s,s),(\epsilon,s)\}$ $E' = \{((u,s),(u',s')) \in (\prec\!\prec_d \cap V' \times V') \mid (s,s') \in E\}$

Figure 19: Upward Transitions between AI and Lithoid Layer

In an upward transition from user lithoid to association layer, a focus in the association layer is defined based on a focus of index expressions (see figure 19).

Thus, the upward transition for users, called `UBeamUp`, is a function from $F_{\mathcal{L}}$ to $F_{\mathcal{L}_{\mathtt{AI}}}$. The set of new nodes V' is constructed out of the set of old ones V by forming these into association nodes. Note that only core nodes (u, u) and user nodes (u, ϵ) are made, since information in user lithoids does not contain information about documents. The nodes that do not belong to the broker's association nodes, i.e., V_b, are filtered out. Once V' is obtained, the new edges E' can be constructed as well. The edges are defined as all possible tuples $((u, s), (u', s'))$, where (u, s) and (u', s') are taken from V', for which the user parts u and u' where connected in the old focus by an edge from E. Disjunction with $\prec\!\!\prec_d$ ensures that only valid association links are obtained.

Downward Transitions
$\mathtt{URestrict} : F_{\mathcal{L}_{\mathtt{AI}}} \mapsto F_{\mathcal{L}}$ $\mathtt{URestrict}(\langle V, E\rangle) = \langle V', E'\rangle$, where $V' = \pi_1 V$ $E' = \{(u, u') \in V' \times V' \mid u \prec_d u' \text{ and } \exists_{s,s' \in \mathcal{L}} [((u, s), (u', s')) \in E]\}$
$\mathtt{SRestrict} : F_{\mathcal{L}_{\mathtt{AI}}} \mapsto F_{\mathcal{L}}$ $\mathtt{SRestrict}(\langle V, E\rangle) = \langle V', E'\rangle$, where $V' = \pi_2 V$ $E' = \{(s, s') \in V' \times V' \mid s \prec_d s' \text{ and } \exists_{u,u' \in \mathcal{L}} [((u, s), (u', s')) \in E]\}$

Figure 20: Downward Transitions between AI and Lithoid Layer

The downward transition, called `URestrict`, is defined in figure 20. The original focus in the association layer is restricted to information about users. Therefore, downward transitions are functions from $F_{\mathcal{L}_{\mathtt{AI}}}$ to $F_{\mathcal{L}}$. The set of new nodes V' consists of all the left hand (user) parts of the old association nodes of V. The right hand parts are simply disposed of, since they do not concern users. Edges are defined between the new nodes: $E' \subseteq V' \times V'$. Every edge in the old focus can lead to an edge in the new focus by stripping of the source parts. To ensure only valid edges, in terms of the direct subexpression relation on index expressions, are obtained, the constraint $s \prec_d s'$ is added.

We now provide some remarks on the definitions just given. When restricting the focus in the association layer to only the user part by a `URestrict` action (see figure 21(b)), all user parts of the nodes are kept and the source parts are thrown away. Note that all source nodes are mapped to the empty descriptor. Edges are only allowed between nodes that are part of the focus. Furthermore, the edges connect user nodes of which the user parts were also connected in the old AI focus. Thus, the structure of the edges in the association index is exploited to form edges in the lithoid layer. In this way, the new focus does not contain new information or knowledge, and, at the same time, keeps as much information as possible.

In beaming up from the lithoid layer to the association index, the nodes of the old focus are translated to association nodes. Thus, the resulting focus of

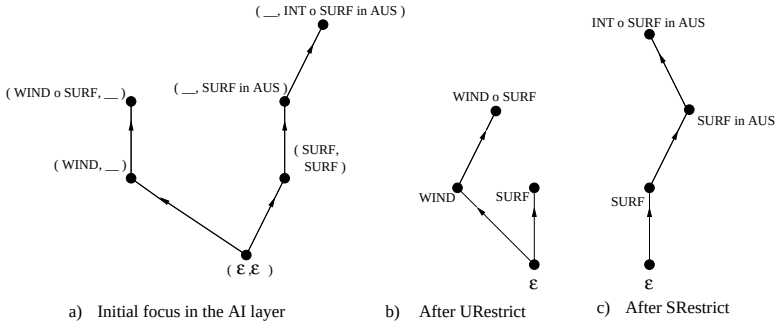

Figure 21: Part a) provides an initial focus in the AI of figure 8, part b) the resulting focus after `URestrict`, and part c) the resulting focus after `SRestrict`.

a `UBeamUp` action contains those nodes of the association index of which the user part belongs to the old focus (see figure 22). These nodes contain extra information, i.e., the source parts, which is based on the knowledge of the broker. The source part is either the same as the user part, resulting in a core node, or the empty index expression, resulting in a user node. This exemplifies the use of broker knowledge in transitions. The links connect the nodes of which the user parts were connected in the old focus. In this way, core links, user links, and user offspring links can be generated, but no invalid or incorrect ones. Note that the definition ensures that the links conform to the association relation ($\prec\!\prec_d$). Furthermore, note that, since the new focus contains no source nodes, source links and source offspring links do not occur in the new focus.

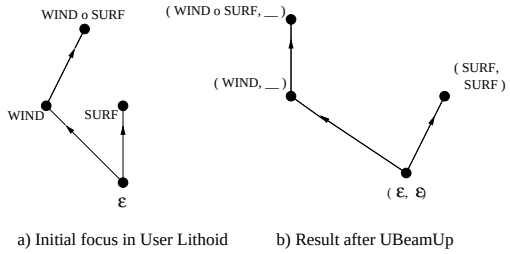

Figure 22: Part a) provides an initial focus in the User Lithoid of figure 3 and part b) the resulting focus after `UBeamUp`.

4.3.4 Properties of Transitions between AI and Lithoid Layer

In this section, we consider properties of transitions between association layer and lithoid layer. In particular, we investigate the sizes of generated foci after transitions between those layers.

Downward Transitions
When restricting a focus in the association index to the lithoid layer, at most as many nodes as in the old focus are generated. Consider the initial focus in the AI of figure 21a. The resulting focus of part b of the figure illustrates that for `URestrict` every source node is projected onto the empty descriptor. The other nodes each correspond uniquely to a node in the new focus. The number of edges in the new focus is at most equal to the number of edges in the old focus in the AI.

Repeated Transitions
In making back and forth transitions between the lithoid layer and the association layer, it is essential where one starts. Starting at the lithoid layer, thus performing an upward transition first, results in an unchanged focus upon return in the lithoid layer. However, starting at the association layer, and thus making a downward transition first, never results in a bigger focus (see figure 23).

Upward First	Downward First
$\forall_{I \subseteq \mathcal{L}} [I = \texttt{URestrict}(\texttt{UBeamUp}(I))]$	$\forall_{T \subseteq \mathcal{L}_{\texttt{AI}}} [T \supseteq \texttt{UBeamUp}(\texttt{URestrict}(T))]$
$\forall_{I \subseteq \mathcal{L}} [I = \texttt{SRestrict}(\texttt{SBeamUp}(I))]$	$\forall_{T \subseteq \mathcal{L}_{\texttt{AI}}} [T \supseteq \texttt{SBeamUp}(\texttt{SRestrict}(T))]$

Figure 23: AI-Lithoid Transition Difference

This difference is caused by the fact that performing a downward transition from the AI to the lithoid layer, inevitably causes a loss of information. This is due to the poorer language of lithoids compared to that of the association index. A consecutive upward transition cannot recover this loss of information since it has no means to identify what information exactly was lost. The information lost in the downward transition is not recoverable by upward transitions.

Over to the other side
Nothing can be said about the size of the resulting focus if one starts at the user lithoid, then beams up to the association index, and finally restricts to the source lithoid. The resulting focus can be smaller, equal, or larger. This also holds for beaming up from the source lithoid to the AI and afterwards restricting oneself to the user lithoid.

5 Applications of the AIA

In this section, applications of the AIA are described. First, retrieval and filtering are expressed in terms of actions in the AIA. Matching, an important concept in IR and IF, is connected with the AIA in section 5.2. Finally, section 5.3 illustrates how several techniques from IR and IF can be incorporated in the AIA.

5.1 Retrieval and Filtering

In the AIA, the basic tasks of information brokers are implemented by series of actions. For each task, there are several different series of actions that implement the task. Note that this conforms to the diversity in techniques to process queries that have been developed in IR and IF systems.

Basic schemes are provided for the elementary tasks of broker agents: query processing and document filtering. The basic query processing scheme consists of creating a new focus based on the query and then performing three transitions. When this is done, the resulting set of documents corresponding to the query is obtained and can be sent back to the user.

The query consists of index expressions. These are found in the user lithoid which implies that the initial focus is in the lithoid layer. In order to obtain relevant documents, the broker agent first has to perform a `UBeamUp` which leads him to the association index. From there, a `SBeamDown` migrates him to the source lithoid, from which a `DocAbout` or `DocId` leads the broker to the desired documents in the base layer. This plan of attack is embedded in the basic query processing scheme below.

Basic Query Processing
`QAccept`(u, Q)
$RelDocs :=$ `DocAbout`(`SBeamDown`(`UBeamUp`(`NewFocus`(Q))))
`Deliver`$(u, RelDocs)$

Here, `QAccept`(u, Q) accepts a query Q from user u, `NewFocus` delivers a new focus, initialized with its argument, and `Deliver` delivers documents to a user. The action `NewFocus` is to be expressed in knowledge maintenance actions of the broker agent. The action `Deliver` is one of interaction with agents of other types and delivers the documents in *RelDocs* to user u.

The basic document filtering scheme is similar to that of query processing:

Basic Document Filtering
`DAccept`$(d, \chi(d))$
$RelUsers :=$ `UserId`(`UBeamDown`(`SBeamUp`(`NewFocus`$(\chi(d))$)))
`SendAll`$(RelUsers, d)$

The `SendAll` command sends document d to all users out of *RelUsers*. The two basic schemes can be refined and augmented in several useful ways. This is the topic of the section 5.3. First, we elaborate on using the AIA to perform matching.

5.2 Matching in the AIA

Another major use of the AIA is for *matching* user interests with contents of offered documents. The AIA is suited for this since it contains both forms of information that are to be matched. Different matching strategies can be

implemented in the AIA. These possibilities can exploit the empty (ϵ) parts of user and source nodes. By filling in these empty places, additional information can be incorporated in tha AIA to support the matching process.

We will now provide two examples of this: contextual matching and preprocessing the association index.

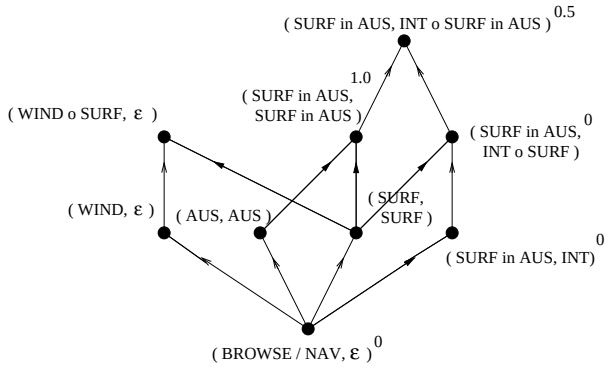

Figure 24: Contextual matching in the AIA

In *contextual matching*, see figure 24, the empty places are filled in by a context. This context, which consists of an index expression, can, for instance, describe a user demand (query) or document content. In figure 24, we consider user context SURF in AUS. A similarity measure computes the strength of the relation between left and right hand side of nodes. The similarity measures identify the part of the AIA that is most relevant to the context. Broker agents can use the similarity measures in selecting links when navigating.

In figure 24, a similarity measure is used that compares index expressions based on their *twigs*. For an index expression I, the twigs are defined as:

$$\tau(I) = \left\{ t_1 c\, t_2 \;\middle|\; t_1 c\, t_2 \preccurlyeq I \right\}$$

The similarity measure, which is a simplification of the pair-wise similarity measure as described in [Ber98], is now defined as the relative overlap in twigs:

$$\mathtt{sim}(I, J) = \frac{|\tau(I) \cap \tau(J)|}{|\tau(I) \cup \tau(J)|}$$

The second form of filling in the empty places in user and source nodes involves *preprocessing* the AIA (see figure 25). No auxiliary information is the preprocessing. The empty places in user and source nodes are filled in by the non-empty index expressions which is the largest (in terms of number of keywords) smaller (in terms of the subexpression relation) one that is part of a core node. In this way, the empty places are filled in by the most expressive index expression that resides in a lower part of the association index. This means that, user and source knowledge are maximally linked.

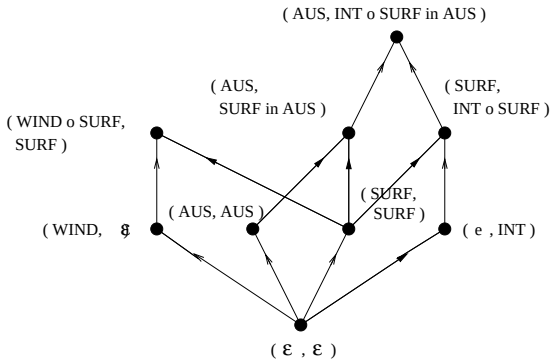

Figure 25: Preprocessing the AIA

The appropriate substituting index expression, is not necessarily uniquely defined. This can occur for user or source nodes that consist of an index expression with at least two terms. In the example, we selected based on alphabetic order. Of course, more advanced selection strategies can be exploited for this.

5.3 Augmentations of Basic Interaction Schemes

This section shows the generality of the AIA by incorporating several useful IR techniques. Since the AIA is a purely symbolical framework, only the symbolical essence of those techniques is incorporated. Further research is needed to refine the AIA with numerical measures. The IR techniques are shown applicable in the AIA by describing them with broker actions.

In the remainder of this section, we touch upon automatic query expansion, different query modalities, query generation, and user profile construction and maintenance.

5.3.1 Automatic Query Expansion

Query expansion (e.g. [MR97] and [AV97]) is addressed as an important issue in (networked) IR. In query expansion, the original query is augmented with a number of new descriptors. These descriptors are taken from a suitable context for the query. Automatic query expansion is an augmentation process of queries where the terms to augment the query with are suggested by the system.

In the AIA, the user knowledge residing in broker agents forms a natural context for queries of that user. Considering a user u having formulated query $Q \subseteq \mathcal{L}$, a broker b can define the context for query expansion in a number of ways:

User Profile Broker agent b can create the context based on the user profile of u, for as far as it is known by b. That is, taking u as initial focus in the base layer, b performs a `Profile` action to obtain the index expressions denoting u's interests. The resulting focus can then be used as starting

point for further processing. This form of query expansion as well as the next on mentioned is a series of (local) navigation and transition actions.

Related Users Broker b can also take interests of related users, which are also known by b, into account. Related users are seen here as users that share the interest in Q. By taking Q as initial focus in the lithoid layer, a `UserAbout` or `UserId` action delivers all the users in the base layer interested in q. By performing a `Profile` action directly afterwards, the total of interests of users interested in q is obtained, which serves as context for query expansion. In this way, a form of social query expansion is obtained.

Related Brokers By communicating with other brokers, broker b can obtain variants of the two approaches given above by using knowledge of other brokers as well. This knowledge can be merged with b's own knowledge. In this process, local navigation and transition actions are combined with communication actions with other broker agents. This will be a topic for further research.

The approach of Fitzpatrick and Dent in [FD97] uses past queries to collect documents for the context, which they call affinity pool. Since brokers are capable of storing past queries in their user knowledge, the AIA can serve as affinity pool as well. Their approach does not require direct user intervention, which is a requirement to automate the process in agents. This also suits query expansion in the AIA. However, their approach uses numerical similarity measures for queries to obtain the best resembling queries and use corresponding top documents only. This also implies the use of a ranked matching mechanism. This numerical aspects are not (yet) modeled in the AIA framework. However, their approach does not distinguish between queries formulated by different users, and thus assumes global common interests, which is not necessary in the AIA.

5.3.2 Query Modalities

Different types or modalities of querying can be identified, which should also lead to different behaviour of the broker agent that receives the query. Modalities of brokering are described in, for instance, [DS97] where roles for broker agents are identified according to initially known capabilities and requests, in [KH97] where matchmaking modalities are expressed in terms of different KQML ([FFMM94]) performatives, and in [WBW98a] where the influence of dual criteria are analysed in the context of information broker design. In addition, different modalities of accessing structured information are described in [WF96]. The different access strategies described there, i.e. fact-finding, learning, gathering, and exploring, can be exploited by broker agents.

Different query modalities demand different processing strategies of broker agents. We already showed several modalities with respect to query expansion. Furthermore, two transitions were defined from lithoid layer to base layer. These can also serve different strategies. Using, for instance, a `DocId` transition, an

overlap measure of relevance for documents is obtained. On the other hand, using a `DocAbout` transition in the same situation results in a more strict containment measure. The former measure suits vague searching tasks as exploring, whereas the latter is more suited for more precise tasks as fact-finding.

Many other query (or document) modalities can be served by variations in the series of actions that implement them. The properties of transitions as stated in sections 4.3.2 and 4.3.4 are useful guidelines in selecting a series of actions.

5.3.3 Query Generation

The use of the term query generation occurs often when referring to translating a query to another form or language. By query generation, we mean from scratch generation of queries by proactive information brokers. The lack of substantial literature on this topic implies that this is a relatively new line of research stemming from the combination of proactive agents and ID.

The user knowledge of brokers, i.e., user profiles, can be exploited for the automatic generation of queries for autonomous (or proactive) IR. The user profile provides a user-set context for query generation. However, for trends and popular or hot topics, additional history data may be needed to generate queries. History data can, for example, deal with the number of times documents are retrieved within a certain period. From this, a generation algorithm like the one described in [MLL92], originally designed to generate graphs, can be constructed. This can serve as a basis for a query generation algorithm. By translating this latter algorithm into navigation and transition actions, it can be adopted for use in the AIA.

Another form of query generation, social query generation, uses communication between broker agents. A broker that receives a query from one of its users, can send this query along to other brokers. These brokers can, in turn, use this query to select documents for their own users. Of course, they can also first adapt the received query to the wishes of their own users by using the user profile (knowledge). This approach may also supply a social form of serendipity by alerting users at articles they did not ask for explicitly but may fall into their interests.

Yet another form of query generation is obtained by a process of adding and deleting terms or parts of an original query. The addition and deletions of terms or larger index expressions coincide with navigation actions in the AIA.

5.3.4 User Profile Construction and Maintenance

The interaction between user, broker, and source agent supplies information the broker can exploit for profile construction and maintenance. According to [WBHW97], three moments for this are identified in an ID application.

First, the queries a broker receives from his users can be used to update the knowledge of the broker, and, by doing that, the user profiles.

Second, at the moment the return set of the query is established, the characterisations of these documents can be used. This can also involve altered characterisations. For instance, if a broker knows that a user is interested in some document and an altered version of this document is received by the broker, the broker can update the interests of the user by including the characterisation of the altered document.

Third, the user can be enabled to specify (detailed) relevance feedback interactively. This information has a high degree of user relevance, since it is directly specified by users. However, direct user intervention is not always wanted or possible.

6 Conclusions & Further Research

6.1 Conclusions

In this article, we proposed the Association Index Architecture (AIA) as a 3-level hypermedia representation. The AIA suits as a representation mechanism for combining knowledge about user interests and offered documents. The AIA is tailor made for information brokers that mediate between users in need of information and sources that offer documents.

The AIA consists of two 2-level hypermedia representations which are connected by the association layer. This association layer comprises a so called the Association Index (AI). The construction of the AI exploits the structured nature of the underlying descriptor language of index expressions. In essence, the AI resembles the lithoids it is based on.

Furthermore, actions were introduced that enable information brokers to use the AIA as a knowledge representation framework. Two kinds of actions were distinguished: navigation primitives and transitions. Our general navigation primitives allow any navigational strategy within a layer to be implemented. Transitions enable information brokers to travel from one layer to another. We provided a number of properties of the AIA and of the actions given.

Using the AIA and available actions therein, information brokers can perform their retrieval and filtering tasks in the context of Information Discovery. They do this by exploiting 2-level hypermedia representations of user interests and document contents. This enables known navigational mechanisms to be used within the AIA framework. We showed how basic query processing and document filtering can be performed in the AIA, as well as matching and more advanced strategies such as automatic query expansion, the use of different query modalities, query generation, and user profile construction and maintenance.

Although the AIA is defined in terms of index expressions, our approach also suits other structured descriptor languages. The only requirement is that the descriptor language can be equipped with a subexpression relation. This holds for every structured descriptor language.

6.2 Further Research

Further research can be directed towards richer descriptor languages than that of index expressions. The language of noun-phrases, for example, which is used in many IR and IF applications, is a natural candidate since it also contains a nicely defined structure. However, this structure is more complex than that of index expressions.

Further research is also needed to further specify and implement matching in the AIA. As mentioned in the previous section, our focus will be on contextual matching and preprocessing the AIA.

In addition, inter-broker actions for sharing user en source knowledge and cooperating based on this should be researched into. This research is more directed to the field of intelligent agents than the augmentations mentioned before. Important issues will be negotiation, for instance in team formation, communication to enable cooperation, and shared knowledge.

Finally, research is conducted into practical issues involving intelligent agents in Information Discovery in the PROFILE project (see e.g. [WSA+97]) of the University of Nijmegen. A generic language for negotiation, called Profile Negotiation Language is built on top of KQML (see [FFMM94]). This language enables more advanced negotiation protocols to be developed quickly.

References

[ACG91] M. Agosti, R. Colotti, and G. Gradenigo. A two-level hypertext retrieval model for legal data. In A. Bookstein, Y. Chiaramella, G. Salton, and V.V. Raghavan, editors, *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 316–325, Chicago, Illinois, October 1991. ACM Press.

[ATKW98] A. T. Arampatzis, T. Tsoris, C. H. A. Koster, and Th. P. van der Weide. Phrase-based Information Retrieval. Technical Report CSI-R9809, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, March 1998.

[AV97] P.G. Anick and S. Vaithyanathan. Exploiting Clustering and Phrases for Context-Based Information Retrieval. In N.J. Belkin, A.D. Narasimhalu, and P Willett, editors, *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 314–323, Philadelphia, Pennslvania, 1997.

[AWBK97] A. T. Arampatzis, Th. P. van der Weide, P. van Bommel, and C. H. A. Koster. Linguistic Variation in Information Retrieval and Filtering. In P. M. E. De Bra, editor, *Informatiewetenschap 1997*, pages 7–10, Eindhoven University of Technology, Eindhoven, The Netherlands, 1997.

[Bat89] M.J. Bates. The design of browsing and berrypicking techniques for the on-line search interface. *Online Review*, 13(5):407–431, 1989.

[BC92] N.J. Belkin and W.B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, December 1992.

[Ber98] F.C. Berger. *Navigational Query Construction in a Hypertext Environment.* PhD thesis, Department of Computer Science, University of Nijmegen, September 1998.

[Bru93] P.D. Bruza. *Stratified Information Disclosure: A Synthesis between Information Retrieval and Hypermedia.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[BW90] P.D. Bruza and Th.P. van der Weide. Two Level Hypermedia - An Improved Architecture for Hypertext. In A.M. Tjoa and R. Wagner, editors, *Proceedings of the Data Base and Expert System Applications Conference (DEXA 90)*, pages 76–83, Vienna, Austria, 1990. Springer-Verlag.

[BW92] P.D. Bruza and Th.P. van der Weide. Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[DS97] K. Decker and K. Sycara. Middle-Agents for the Internet. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, August 1997.

[FD97] L. Fitzpatrick and M. Dent. Automatic Feedback Using Past Queries: Social Searching? In N.J. Belkin, A.D. Narasimhalu, and P Willett, editors, *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 306 – 313, Philadelphia, Pennslvania, 1997.

[FFMM94] T. Finin, R. Fritzon, D. McKay, and R McEntire. KQML - a language and protocol for knowledge and information exhange. In M. Klein, editor, *Proceedings of the 13th International Distributed Artificial Intelligence Workshop*, 1994.

[Hui96] T.W.C. Huibers. *An Axiomatic Theory of Information Retrieval.* PhD thesis, Department of Computer Science, Utrecht University, November 1996.

[Joh95] P. Johannesson. Supporting Schema Integration by Linguistic Instruments. In *Proceedings of the First International Workshop on Applications of Natural Language to Databases (NLDB'95)*, pages 41–56, Versailles, France, June 1995.

[JSSW95] A. Jameson, R. Schäfer, J. Simons, and T. Weis. Adaptive provision of evaluation-oriented information: Tasks and techniques. In C. S. Mellish, editor, *Proceedings of the IJCAI-95 Conference*, pages 1886–1893. Morgan Kaufmann, 1995.

[KH97] D. Kuokka and L. Harrada. Matchmaking for Information Agents. In Huhns M.N. and M. Singh, editors, *Readings in Agents*, San Francisco, California, 1997. Morgan Kaufmanm Publishers.

[MLL92] S.H. Myaeng and A. Lopez-Lopez. Conceptual graph matching: a flexible algorithm and experiments. *Journal of Experimental Theoretical Artificial Intelligence*, 4:107–126, 1992.

[MR97] M. Magennis and C.J. van Rijsbergen. The potential and actual effectiveness of interactive query expansion. In N.J. Belkin, A.D. Narasimhalu, and P Willett, editors, *Proceedings of the 20th Annual International ACM SIGIR Confe rence on Research and Development in Information Retrieval*, pages 324–332, Philadelphia, Pennslvania, 1997.

[MRF+90] G.A. Miller, Beckwith R., C. Fellbaum, D. Gross, and K.J. Miller. Introduction to wordnet: An on-line lexical database. *Journal of Lexicography*, 3(4):234–244, 1990.

[Ric79] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

[Rij75] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, United Kingdom, 1975.

[RIS+94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, NC, 1994. ACM.

[Sim97] J. Simons. Using a semantic user model to filter the world wide web proactively. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *Proceedings of the sixth international Conference UM97*. SpringerWienNewYork, 1997.

[SV97] T. Sandholm and Lesser V. Coalitions among Computationally Bounded Agents. In *Artificial Intelligence*, volume 94, pages 99–137, 1997. Special ssue on Economic Principles of Multiagent Systems.

[WBHW97] B.C.M. Wondergem, P. van Bommel, T.W.C. Huibers, and Th. van der Weide. Towards an Agent-Based Retrieval Engine. In

J. Furner and D.J. Harper, editors, *Proceedings of the 19th BCS-IRSG Colloquium on IR research*, pages 126–144, Aberdeen, Scotland, April 1997.

[WBHW98] B.C.M. Wondergem, P. van Bommel, T.W.C. Huibers, and Th.P. van der Weide. Agents in Cyberspace – Towards a Framework for Multi-Agent Systems in Information Discovery. In *Proceedings of the 20th BCS Colloquium on Information Retrieval, IRSG98*, Grenoble, France, 1998.

[WBW98a] B.C.M. Wondergem, P. van Bommel, and Th.P. van der Weide. Cumulative Duality in Designing Information Brokers. In *Proceedings of the 9th International Conference and Workshop on Database and Expert Systems Applications*, Vienna, Austria, August 1998.

[WBW98b] B.C.M. Wondergem, P. van Bommel, and Th.P. van der Weide. Formalisation of Index Expressions. Technical report, University of Nijmegen, Nijmegen, The Netherlands, June 1998. Technical Report: CSI-9817.

[WF96] R. Wilkinson and M. Fuller. Integrated Information Access via Structure. In M. Agosti and A. Smeaton, editors, *Hypertext and Information Retrieval*, pages 257 – 271, Boston, U.S.A, 1996. Kluwer.

[Win83] W. Winograd. *Language as a Cognitive Process*. Addison-Wesley Pub. Co., Reading MA, USA, 1983.

[WJ95] M. Wooldridge and N.R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[Won96] B.C.M. Wondergem. Preferential Structures for Information Retrieval. Master's Thesis INF-SCR-96-21, Department of Computer Science, Utrecht University, Utrecht, The Netherlands, August 1996.

[WSA+97] B.C.M. Wondergem, J. Simons, A.T. Arampatzis, J. Mackowiak, D. Tarenskeen, and T.W.C. Huibers. Profile Information Filtering Project – Overall Project Plan. Technical report, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, September 1997. CSI-N9707.