Dagobert Soergel
College of Library and Information Services
University of Maryland
College Park, MD 20742
Office: (301) 405-2037 Home: (703) 823-2840
Fax: (301) 314-9145 dsoergel@umd.edu www.dsoergel.com

A general model for searching linked data
or
Design of an integrated information structure interface

A unified framework for indexing and searching

in database, expert, information retrieval, and hypermedia systems

UB LIS 571
Lecture 4.1, Reading 1

January 1999, edited with a new title December 2011

Design of an integrated information structure interface

Abstract

This paper presents a general information structure model and an interface design that allows users to search and interact with the structure. Both draw power from simplicity. The objective is to develop an intuitive information structure model that is immediately obvious to the user. The unified information structure model integrates information retrieval, hypermedia systems, database systems, and expert systems. The user of such a system accomplishes hypermedia navigation, retrieval, and inference all with the one search syntax that is as simple and natural as it is general and powerful. The paper demonstrates the essential unity of database, expert, information retrieval, and hypermedia systems and discusses the dimensions of a design space in which specific systems can be located. The unified view is developed through a borrow-and-generalize strategy: Find an example from one context, abstract a principle from the example, and apply the principle in another context. Many examples illustrate this approach.

The emphasis of this paper is to describe an information structure and interface intuitive for a user. This approach can be implemented through an interface providing unified access to many heterogeneous systems, all mapped to same model, or through a system that implements a unified information structure in its storage of data..

The paper describes the basic structure, a plain entity-relationship model without the complication of attributes; binary relationships are also called links. Collections or sets of objects, called neighborhoods, provide more structure and play a key role in searching. The paper then presents a classification of search types and search metaphors, including the navigation metaphor – use a known starting point to follow links to unknown but presumably helpful target objects – and query-based search. Specifying two starting points such that a target is retrieved only when it can be reached from both is a natural way of specifying Boolean AND. Starting from a whole neighborhood of objects is a natural way for specifying Boolean OR. Specifying whole neighborhoods of objects as search targets is a new and very powerful element of our design. For example, the user can specify single paragraphs, whole documents, or the neighborhoods of documents by the same author as targets for an AND search. Hierarchical inheritance is an integral part of the model.

Part 2 discusses a number of advanced concepts and applications, including modeling a programming environment and the use of data about the structure, or metadata.

Part 1. Basic structure and search commands

Part 2. Extensions and refinements

0 Prolog: Finding answers. The nature of search

To serve a user confronting a problem, bring her

from her present state of knowledge

to a state of knowledge that enables her to solve the problem.

There are many types of answers and many processes that lead to answers. Answer-finding processes include the system operating on stored data and the user interacting with the system.

Examples

1 The user enters a problem (such as a medical diagnosis problem or a computer system configuration problem) to an expert system and the system, drawing on a vast body of knowledge and many inferences, provides an answer. The answer may be just one solution, or a list of solutions in rank order by likelihood (for the diagnosis) or goodness of fit (computer system configuration).

2 The user enters a request for a statistic, such as the median income in Washington, DC. The system processes census data and gives the figure.

3 For marketing purposes, the user asks for a list of ZIP code areas in the Northwestern US with a median income above $40,000. The system computes the median income for all applicable ZIP code areas and presents a list of those that meet the criterion.

4 The user, a faculty member, asks for a list of students who are enrolled in one of her current course offerings, giving the course offering ID. The system compiles the list through straight one-step retrieval. Variation: The user asks for lists for all her current course offerings. The system uses two-step retrieval: from faculty member to course offerings, from each course offering to students.

5 A user wants to explore Greek culture. He logs on to Perseus and starts at a place he knows, say Sparta. Goes to buildings there, finds a theater, looks at pictures of that theater, does a search for other theaters, finds an encyclopedia article about Greek theaters, etc. This search proceeds in many steps; in each search step the user selects a starting point and the system finds one or more items related to the starting point through a link in the system's database. In each step the user learns a little bit of what he wants to know, cumulatively he learns a lot. (The "berry-picking" approach to finding an answer, Marcia Bates)

6 A user retrieves a known relevant document by author and then asks the system to find similar documents (assuming the system has such a command). Note that there are degrees of likeness and that are many criteria to judge likeness.

7 A sociology sophomore with rudimentary knowledge of statistics but no knowledge of

calculus wants to learn more about statistics, particularly non-parametric tests. She uses a hypertext statistics text. She looks for an introduction to non-parametric statistics, find that she does not understand it, and follows a <*hasPrerequisite*> link to acquire some necessary background. While reading this material, she needs some examples to understand it better. Of several links to examples she chooses the one marked *sociology*. The student thus navigates her way through the hypertext and eventually learns what she came to learn.

8 Another student in the same situation uses a different interaction with the system. He tells the system what he wants to learn and answers question about his background knowledge and subject field. The system then uses its knowledge of the prerequisite structure to chart a path through the system, choosing examples from sociology. The system creates a made-to-order textbook for the student.

9 The user searchers AltaVista with a three-word query, looks at the first ten documents (ranked by AltaVista according to expected relevance), examines document 6, finds a link to a Web site that happens to be a reference site for her topic, and from there finds three really useful documents.

10 A user signs on to a system with information about all kinds of items for sale – houses, cars, boats, computers – for sale to look for a house but does not really know all the questions to ask. In other words, the user does not understand his problem completely. The system helps about by presenting an introduction to house buying structured as a hypertext and a checklist of things to watch out for and information to collect (from the system and elsewhere).

11 A researcher with moderate knowledge of statistics is looking for a software package to do statistical analysis. He consults a software database that evaluates each type of software using a standard set of evaluation criteria; they have a frame with slots to be filled. The system uses the same frame to elicit the users need; for each evaluation criterion (each frame slot) the user enters a weight. For example, this user would put a high weight on the programs ability to provide explanations under what conditions a statistical test is appropriate. The system then returns a ranked list of programs, taking into account the users weights on each criterion and the program's score on that criterion.

12 A user does a search in OPAC for *concept*. She finds relevant and not so relevant documents. Among them is one titled *Concepts and categories: philosophical essays*. The user realizes that *categories* is also a good search term. Using it she finds *George Lakoff. Women, fire, and dangerous things. What categories reveal about the mind*. This is **relevance feedback** executed by the user.

13 A user searches a more sophisticated retrieval system. This system uses a weighting scheme much like the software retrieval system in example 8 and returns a ranked list of documents. The user indicates which of the retrieved documents are relevant. Now **the system** looks at all the terms occurring in the relevant documents, increases there weight in the query formulation, and then repeats the search with the modified query formulation, producing a ranking of documents that is closer to what the user had in mind. This is relevance feedback executed by the system.

14 A student looking for information for a term paper has a general idea that her topic is in library automation; she has an inkling that may be she is interested in writing on how library automation helps users, but does not know how to express that thought. So she enters a broad query to a bibliographic retrieval system and gets back many documents, but the display has a twist: it is a semantic map that groups similar documents together and shows the thematic structure of the topic. The student might ask for a listing of titles in the areas *services*, *search*, *user*, and *impact*.

Map display from Xia Lin

Principles

What do all these examples have in common? A search consists of one or more search step. Each search step starts from something known and a type of link that are both indicated by user and lead to something (or specify something) wanted but unknown; the system finds or creates what is wanted. In a one-step search the user must come up with a very complete specification what is wanted, a very complete list of items known that specify what is wanted but unknown. The system then goes to work to use this clues and comes back with the final result. In a multi-step search the user's idea of what is wanted develops gradually in many small search steps, none of which require high cognitive effort on the part of the user in coming up with the search specification nor a lot of work from the system in finding the intermediate result wanted. Searches differ in the distribution of work among search steps.

The examples also illustrate that there are different ways of specifying the starting point – through remembering and entering a value, such as a subject descriptor, or through clicking on a item displayed on the screen.

A further aspect is the division of labor between user and system. In some examples the system comes up with a specific result, but in example 13 the system comes up only with a broad selection form which the user must further select.

In the specific case of a search for relevant documents (or programs, or people), retrieval may mean simply to find the documents that meet the requirements of a Boolean query; this is the case with most retrieval systems in use today. But in the broadest sense, **retrieval** means the following: **The system should use all available evidence to predict the degree of relevance of an entity (document, program, person, etc.) to the particular user in the particular situation**. In bibliographic subject retrieval such evidence might include: Subject descriptors assigned to the document (if any), words in the title, words in the abstract and text of the document, possibly considering the number of occurrences of the word in the document compared to the collection frequency, the syntactic relationship of the words in the document, relationships between words given in a thesaurus, the author and his or her subject area / area of expertise, the journal where published, the rating of the journal.

1 Introduction. Scope, purpose, and organization of the paper

1.1 General introduction and executive summary

This paper presents a general information structure model and an interface design that allows users to search and interact with the structure. Both draw power from simplicity. The objective is to develop an intuitive information structure model that is immediately obvious to the user.

Database systems, expert systems, Information Storage And Retrieval (IR) systems, and hypermedia systems, all have their own functionality and strengths. Yet on an abstract level these systems have more in common than meets the eye, and functions that on the surface seem different are in fact essentially the same. Consider intelligent databases vis-a-vis expert systems. Different functionalities also complement each other as demonstrated by the use of hypermedia approaches in the interface to expert systems.

What if we could design a system that would draw on the commonalities, combine and integrate all the different functions, and through the integration enhance them, offering great power with an intuitive interface? This is exactly what this paper begins to do. It sets out to demonstrate the essential unity of these different systems and to develop a unified framework for improved design. It sets out to define a system structure and generalized search and inference operators that are applicable to any type of searching – by subject, by citation, by prerequisite, or by any other criterion, following any kind of link – and thus make searching both simpler and more powerful. We use the term **Information Structure Management** to emphasize the key element of structure in common to all types of systems encompassed in the unified approach. The model can be implemented through an interface that maps from several underlying databases, each with its own structure, but in a manner that is transparent to the user; the user would deal only with the one unified view of data. Alternatively, the unified model could be used as the database structure, in which case one could use the term Information Structure Management System.

This unified information structure model integrates the functionality of all types of information systems. Thus a user can accomplish hypermedia navigation, retrieval, and inference all with the same search syntax that is as simple and natural as it is general and powerful.

To develop a unified view is not to ignore the differences in functionality between different types of systems. The systems do indeed differ along a variety of dimensions, but these differences are only a matter of degree. Each type of system, and each individual system, can be located in a multidimensional design space, and doing so improves our understanding of these systems. Most present systems offer the user one type of functionality represented by a fixed location in the design space. The unified view envisions a system where the user has a choice of multiple functionalities and can combine different functionalities.

The basic structure is a plain entity-relationship model without the complication of attributes:

- It consists of objects (entities) and relationships (in the special but frequent case of binary relationships: links) that are used to make statements.

- The basic search operation is also very simple: Specify a starting point and a link type, and the system will follow all links of that type to reach targets that presumably fill the need.
- A search using a chain of links effects spreading activation as well as inference. (A named chain of links, each link of a given type, is equivalent to a Prolog rule.)
- Specifying two starting points such that a target is retrieved only when it can be reached from both is a natural way of specifying Boolean AND. (This could be used for finding all documents indexed by two descriptors, or all descriptors used in the indexing of two known relevant documents, or all Web pages with one- or two-step links from both of two known Web pages.)
- Starting from a whole "neighborhood" of objects is a natural way for specifying OR.
- Specifying whole neighborhoods of objects as search targets is a new and very powerful element of our design. It offers the user great flexibility in doing searches that exploit the information given by all the links in the database, including the specification of hierarchical inheritance. It lets the user specify the target scope of a Boolean AND or spreading activation search, requiring co-occurrence in a sentence, or a paragraph, or a chapter, or a whole document. It also lets the user control the granularity of the items retrieved.

The paper will illustrate these principles through numerous examples from many different contexts. In particular, we will use a borrow-and-generalize strategy in developing and presenting the unified approach. Find examples from one context and then generalize the principle to other contexts. For example, consider Boolean searching which is common in IR systems. It might be quite useful in hypermedia systems to extend navigation, which leads from a starting document to a citing document or a supporting document, to a Boolean search, where the user could start from two documents and look for documents that cite **both** (known as cocitation) or that support **both**. The examples have been chosen with a view to demonstrating the richness and power of the proposed structure.

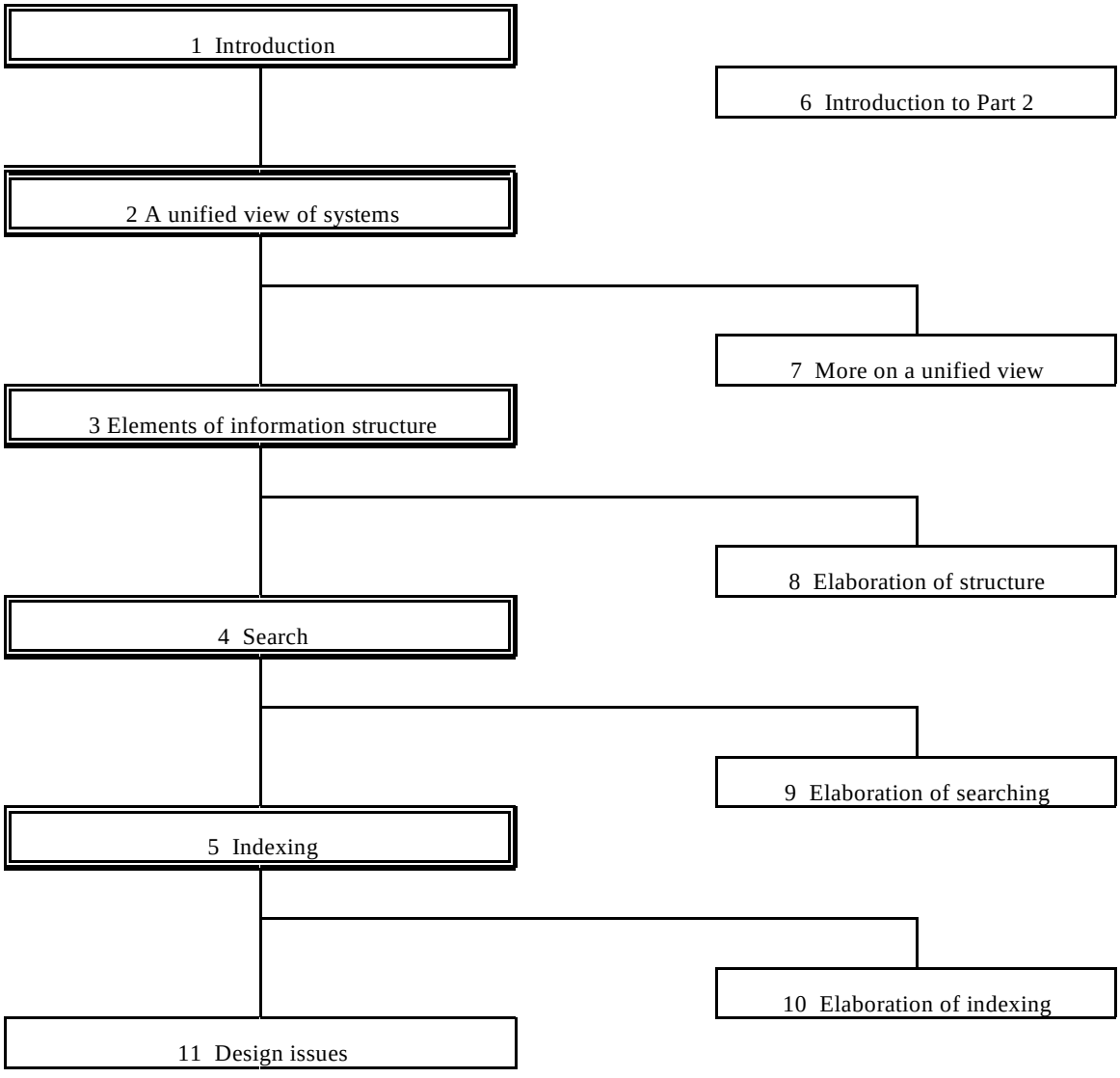
Many of the ideas presented have been implemented in one context or another. The contribution claimed is the unified view and the resulting transfer of techniques from one context to another, especially the powerful concept of specifying arbitrary neighborhoods as search targets, resulting in a general, integrated powerful yet intuitive system.

1.2 Organization of the paper

The paper is divided into two parts. Part 1 describes the basic ideas, providing just enough detail and context. Part 2 adds many interesting elaborations and refinements that further illustrate the power of the approach. Figure 1 shows the dependencies among the sections; the Sections in Part 2 repeat the sequence of Part 1; the sections in Part 2 can be read independently of each other. A reader who wants to thoroughly explore each aspect of the proposed model before going on could read the sections in the sequence 1 - 2 - 7 - 3 - 8 - 4 - 9 - 5 - 10 -11.
Part 1 starts out with an introductory example to provide a backdrop for the more abstract development of the information structure and search model (Section 1.3). Next it shows that distinctions often made, particularly the distinction between traditional (bibliographic) retrieval

systems and typical hypermedia systems, are not absolute but a matter of degree in a multidimensional design space (Section 2). (The perception of these distinctions as absolute would interfere with an understanding and appreciation of the unified approach .) A general structure and data model for an integrated information structure interface (Section 3) provides the basis for discussing search (or navigation) (Section 4) and system construction or indexing (Section 5). As already mentioned, Sections 6 - 10 in Part 2 augment Sections 1 - 5 of Part 1. Finally, Section 11 considers some issues of system design, especially building a user interface that would provide the power of the general approach without overwhelming the user.

Figure 1. **Dependencies among sections** (Double-frames designate core sections.)



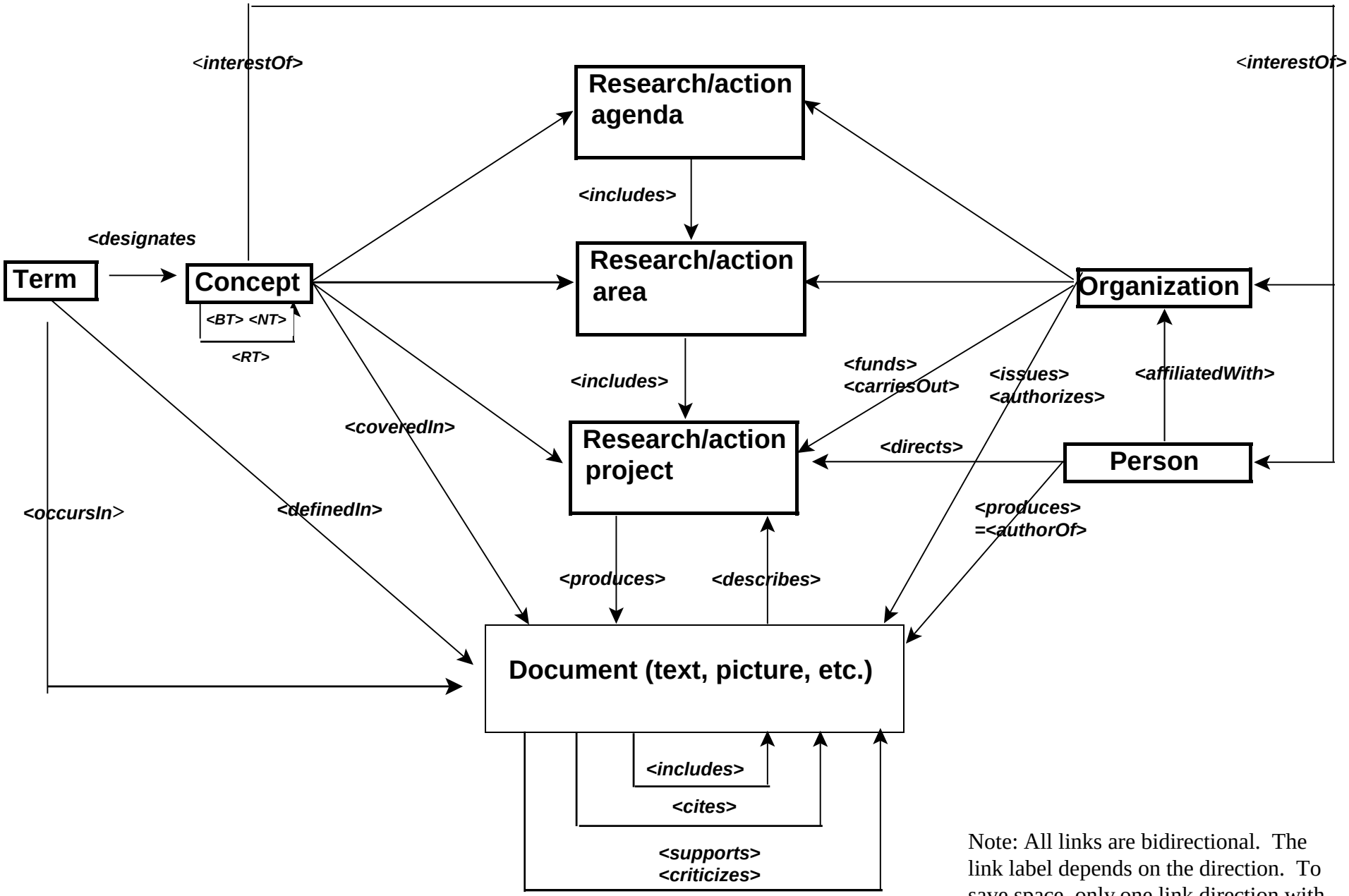
1.3 **Introductory example:** Concepts, projects, texts, organizations, persons

The example presents the design of an augmented digital library that uses a rich set of links between a variety of object types. The example sets the stage for working out the principles of the integrated information structure interface in Sections 3 and 4 and provides experiential background for a better understanding of these principles.

Figure 2 presents the conceptual schema of the hypothetical digital library. Figures 3a-c give an example of a simple subject search, Figure 4 gives an example of a typical hypertext interaction; both examples use the same interface and navigation metaphor. Figure 3a illustrates a search for a single concept in a screen mockup. The single concept, represented as a box, is expressed as an OR combination of descriptors from the Alcohol and Other Drug Thesaurus, making use of the descriptor relationships from the thesaurus. Figure 3b presents a search for two concepts ANDed. Figure 3c gives a brief description of the envisioned interface and shows how it uses the structure.

A search might start with the presentation of the information structure, the system's conceptual schema, in an easy-to-grasp graphical format so that the user has a concrete image of the search possibilities and can exploit this to map her need to the system. One could show the user the whole structure in a large scrollable window, use zooming to show only a segment of the total structure, or let the user select a target object type (from a list or typed in and mapped to the system terminology) and then show the part of the structure surrounding that object type.

The idea is best illustrated with a sample interaction (See Figures 2 and 3a and 3b) and the example of a user-system dialog following them.



Note: All links are bidirectional. The link label depends on the direction. To save space, only one link direction with the corresponding label is included.

Figure 2. Portion from an information structure schema

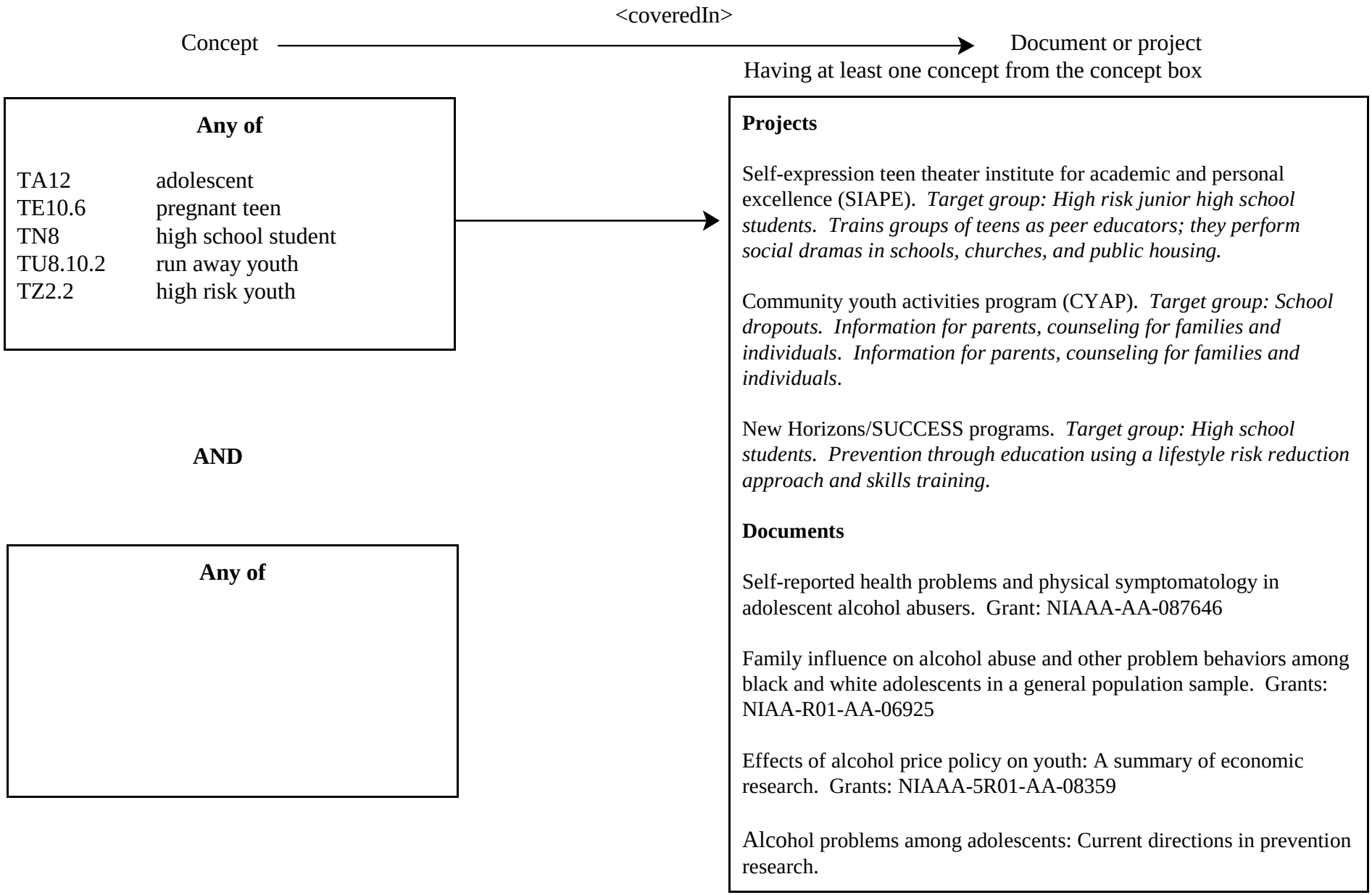


Figure 3a. Search for the concept "adolescent" (enhanced with further concepts from the AOD Thesaurus).

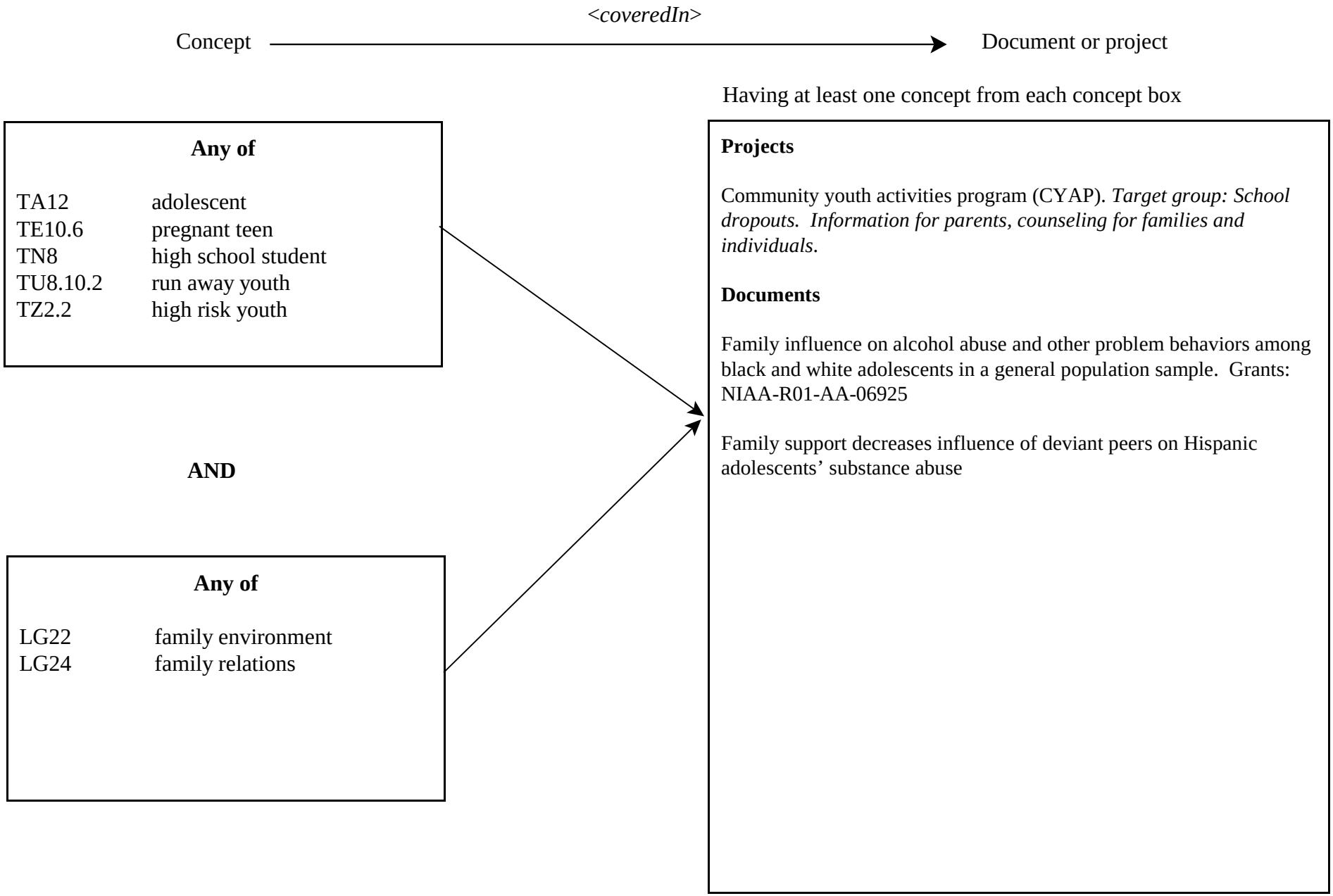


Figure 3b. Search results after ANDing a second concept related to *family*.
Figure 3c. User-system dialog example

This interaction guides the user through a Boolean query formulation in terms that are natural. Boolean OR and AND are visualized through two different graphical constructs: OR through a box, AND through the arrows that signify *reachable from both starting points*. (In terms of Section 4, boxes are used to delineate neighborhoods, and an OR search is a from-neighborhood search).

System	*Click on what you want to find (document, research project, person, etc.)*
User	After studying the schema, clicks on *document* and *research project* (If the same search pattern, such as a subject search, is to be used for two target object types, the user can combine two searches in one by indicating two target object types as in the example; it is quite natural for a user to request both documents and research projects on a topic).
System	The frames for target object types become red. *Click on what you want to start from (for example, searching for documents starting from concept or starting from person)*
User	Clicks on *concept* (to search by concept or subject)
System	The concept frame becomes green. All possible chains of links from the starting point to the target are highlighted in yellow. The links become one-directional on a chain from starting point to target. *Click on the links to be followed from starting point to search target*
User	Clicks on the <*coveredIn*> links from *concept* to *document* and from *concept* to *research project*
System	The search links turn green, the general structure of the search is now visible A screen as in Fig. 2a with blank boxes appears (an enlarged segment of the structure). It would look slightly different from the figure, preserving the colors and labels from the overall structure diagram. The **Any of** in the box itself and a message on the bottom of the screen indicate the meaning of Boolean OR (without actually using the term)
User	Enters one or more values in the starting box.

System	Suggests further starting values to be entered in the box, for example, narrower concepts from a thesaurus, asking the user to select
User	Accepts none, some, or all of the suggested concepts
System	Fills the result box with the target objects that can be reached from **any of** the starting points through the specified link chain(s). *Do you want to restrict your search further by requiring a second concept yes/no* (or whatever the starting object type is)
User	Clicks *yes*
System	A second starting box opens up, with a link arrow to the result box, indicating **all of** where the two arrows join and explaining to the user in a message on the bottom of the screen that it will show only target objects that can be reached from **both** starting points.
User	Enters one or more concepts into the second **Any of** box Similar interaction as before or a more complex interaction using some feedback mechanisms, such as giving a list of all descriptors occurring in objects shown in the result box, indicating their frequency, having the user indicate some relevant objects in the result box and derive descriptor suggestions from there, etc. This interaction continues until the user is satisfied with the terms in the second start box.
System	Shows in the result box only those objects that can be reached from both starting points.

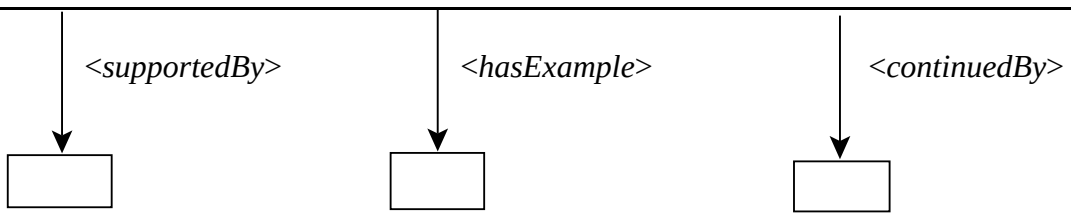
Figure 4. Example of a typical hypertext interaction

The user clicks on the first document in Figure 3b and is presented with an outline constructed from *includes* links:

Barnes, Family influences on alcohol abuse and other problem behaviors among Black and White adolescents in a general population sample. 1995.
METHOD Sampling. *Retention* Measures. *Dependent Measures for Adolescents* / *Independent Measures for Parents* / *Peer Variables* / *Other Independent Variables* RESULTS Descriptive Analyses. *Sociodemographic Factors* / *Adolescent Behaviors* / *Parent and Peer Factors* Multivariate Analyses DISCUSSION

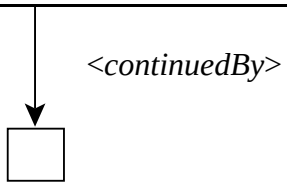
She clicks on the heading DISCUSSION and is presented with the first paragraph

Barnes, Family influences on alcohol abuse and other problem behaviors among Black and White adolescents in a general population sample. 1995. DISCUSSION
The central message from our study is that the quality of parenting is critically important for adolescent outcomes regardless of race or other sociodemographic considerations. High levels of parental support and monitoring, as well as positive adolescent-parent communication, are therefore key elements in the prevention of alcohol abuse and other deviant behaviors. It is important to note that the parenting variables used in this analysis are adolescents' perceptions of their parents behaviors. Using cross-sectional data from Wave 1, we have reported elsewhere that mothers' reports on the quality of parenting are generally higher than adolescents' reports--that is, mothers' mean levels of support and control are higher than adolescents' mean levels. Nonetheless, both adolescents' reports and mothers' reports show the same significant impact on adolescent behaviors. (See Barnes & Farrell, 1992).



The user looks for other evidence supporting this finding and clicks on the link <*supportedBy*> (or the box it leads to) and is presented with a paragraph from another document:

Frauenglass, Family support decreases influence of deviant peers on Hispanic adolescents' substance use. *Journal of Clinical Child Psychology*, 1997. DISCUSSION
Although adolescents spend twice as much time with their peers and at school than with their family (Berger, 1994), the results of this study indicate that parents are by no means defenseless against the powerful influence of peers in their adolescent's life. Family support as reported by the adolescents accounted for a significant amount of unique variance in the prediction of adolescent alcohol use, after controlling for all the *peer deviance variables*. As the level of perceived family support increased, self-reported adolescent alcohol use decreased. Although family support did not have statistical significant main effects in predicting adolescent tobacco and marijuana use, the interactions of family support with peer tobacco and peer marijuana use were significantly related to lower self-reports of tobacco and marijuana use. For both of these substances, when the number of substance using peers increased, higher levels of family support were associated with reduced levels of adolescent substance use.



The user clicks on the highlighted *peer deviance variables* to get an explanation:

Frauenglass, Family support decreases influence of deviant peers on Hispanic adolescents' substance use. *Journal of Clinical Child Psychology*, 1997. MEASURES
Peer deviance variables Self-report on how many of their friends used tobacco alcohol marijuana cocaine Self-report of how many friends belonged to a gang Responses: *none, 1, 2, 3-4, 5 or more*

The user then returns to the Barnes DISCUSSION frame and clicks on the <*hasExample*> link:

Foxcroft, Role of the family in adolescent alcohol abuse: Socialization and structural influences. *Journal of Adolescent Chemical Dependency,* 1992.

To illustrate the influential role of family socialization factors and family structure on drinking behavior, two example case studies are described. Both are taken from a larger, ongoing study.

Case Study 2

To contrast with Tony' history, we describe below the development of Darren's drinking behavior – which could be regarded as "sensible"

Case report

Darren is a 17 year old apprentice welder. He lives at home with his parents, younger sister, and younger brother. Darren speaks warmly of his family. Darren's father is a crane driver, and his mother has had various part-time shop assistance jobs. Darren's father has never been out of work, and his mother stayed at home to look after the children. Darren's mum and dad get along well. Darren's mother goes to church regularly, but there is no pressure to conform to any religious viewpoint.

Right from the outset we can see that Darren described his family in a positive way – he speaks warmly of his family, his parents get on well, and there is no pressure to conform to his mother's religiosity. This suggests a *supportive and democratic family environment.*

<*continuedBy*>

The remainder of the case report describes Darrel's internalized norms and the methods of support and discipline his parents used

Figure 5 on the facing page shows how a citation search using AND (finding documents that include both of two known relevant documents) can lead to further highly relevant documents.

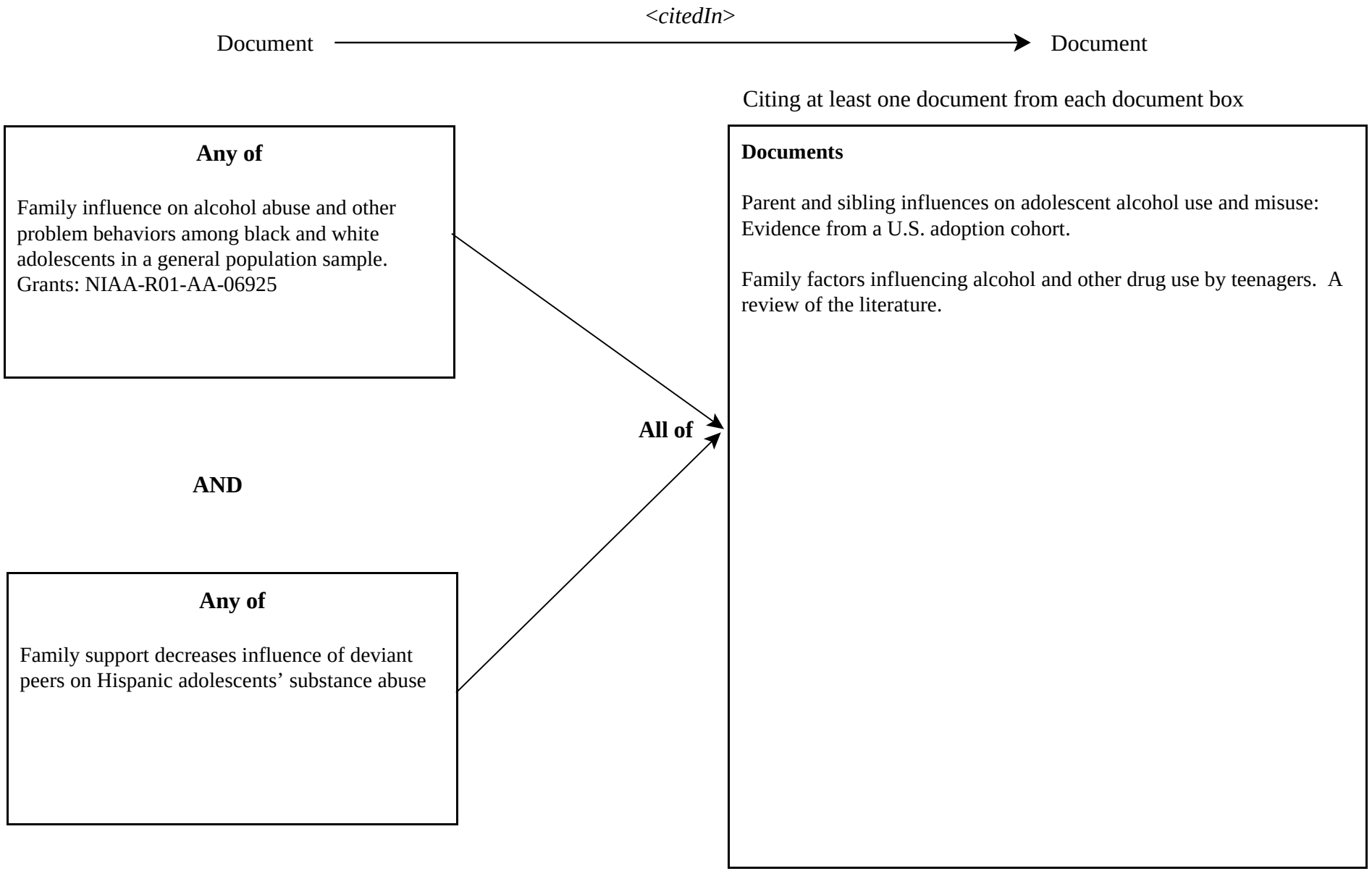


Figure 5. Citation search using AND

2 A unified view of systems or The multidimensional design space for information systems

This section concentrates on working out the dimensions along which the typical hypermedia system and the typical bibliographic retrieval system can be compared, crystallizing both differences and commonalities. Relationships among these and other types of systems – database management systems and expert systems – will become clear from the examples in other sections of the paper.

Many hypertext or hypermedia systems offer users two systems of access which are usually presented as distinct and different: Access through navigation following links and access through index search. But in principle there is no difference between starting from a known paragraph of text and following links to find one or more unknown paragraphs and starting from a known term and finding one or more unknown paragraphs. Rather then burdening the user with two different methods of searching, systems should offer a unified interface.

On the other hand, there are differences between types of searches and, by implications, between systems. These differences involve several independent dimensions, and they are differences of degree rather than either-or. Analyzing these differences is important as a basis for system design (Figure 6). Figure 7 gives a summary of the most important of these dimensions; Section 7 gives more detail.

Figure 6. **Principles for analyzing and comparing searches and systems**

"Traditional" information retrieval systems and hypermedia systems are designed to support certain kinds of searches.

The differences can be analyzed along a number of dimensions shown below.

These differences are a matter of degree.

An integrated information structure interface should support all types of searches, adapting to user needs and preferences.

A few introductory examples are useful as a backdrop for the general discussion. When I start preparing a lecture on a new topic, I might first do a literature search, starting with a suitable subject descriptor that leads to a list of documents and pick a document to read first, or I might remember a document that would give a good start. The starting document might lead to a document cited, or it might teach me that I must find out about another topic first and do another search; if there are no such leads, I simply proceed to another document from the retrieved list.

The reader of a book might follow the Red King's advice: "Begin at the beginning and go on till you come to the end: then stop", reading one chapter after the other, one paragraph after the other. Or she might deviate from the author's sequence, consulting, for example, the table of contents or the diagram showing the logical dependencies between chapters as seen in Figure 1 or found in the front of some books, especially in mathematics. Or she might deviate from the linear path by looking at some of the footnotes or by following a cross-reference.

Some people may read the newspaper from beginning to end, but most of us pick certain sections, scan the headlines, read the first paragraph of a story and perhaps go on in that story, or jump on to the next headline. In an electronic newspaper the reader can enter an interest profile, a set of terms, as a starting point from whence a search leads to a list of stories or articles in order of importance or grouped by subject.

In each of these examples, the search for information is a journey that evolves in a series of steps. Each step starts from a point that is known and moves to a new point that is yet unknown. The user must find the next stop on the journey, the item of information to be consulted next; an IR/hypermedia system must support the user in this retrieval task. The term "user" is used here in the broadest sense, including searcher and reader.

An item of information can be any text object – from a book to a sentence – or any other media object, such as a picture, a map, or a segment of sound or a moving picture sequence that can be replayed. The term "Media object" is a broad term that covers all of these; specifically, many media objects consist only of text ("Text objects") or contain a large proportion of text. For simplicity, we will generally use the term "Document" in the broadest sense of media object; in some contexts, document refers specifically to a text object. When referring to a book or journal article or other document in the more narrow traditional meaning, we will generally use the phrase "whole document" or "document as a whole". The general framework presented here deals more broadly with any type of object, entity, or item. An object is also called a "node" when the emphasis is on the place or role of the item in a network of links. In the hypertext literature a document is also called "frame" (in the sense of all the information fitting on one screen, not to be confused with frames used for knowledge representation), "notecard", or simply "card".

Reading a whole document, such as a book or journal article, is a special case of search: The question on what chapter (or section, or paragraph) to read next is seldom posed explicitly; by default the reader goes on to the next chapter (or section, or paragraph). But there exists a retrieval problem nevertheless, and a book solves this through its arrangement, its table of contents or an outline in flow chart form as mentioned above. A book that is available as a hypermedia base makes the user (reader) aware of choices and suggests detours or radical deviation from the default path. It does so by supporting retrieval through a network of links among paragraphs (or sections, or chapters) that allow the reader to select (or retrieve) another section starting from a section just read. This kind of retrieval is an integral part of reading. Granularity, the size of the nodes used, differs from system to system; a hypermedia system could treat an entire journal article or even an entire book as a unit linked to other units. For example, a database of full-text journal articles in which the user can jump from a reference in the

bibliography of article 1 directly to the referenced article 2 (as offered by INSPEC) is quite useful; such a database is feasible, while transforming all these articles into a fine-grained hypermedia base would involve a huge effort.

When the user is faced with the question of what entire document to read next in his quest for information, he turns to a bibliographic IR system. There he cannot start from a paragraph he just looked at; most systems require a query formulation as a starting point; a query formulation contains subject descriptors or author names, etc.

The basic retrieval operation is always the same: Starting from a known object – a document or a descriptor – the user follows links provided by the system to find one or more other objects. On an abstract level, using a known document as a starting point to find descriptors for further searching is the same thing as using a known descriptor as a starting point to find documents for perusal. Even Boolean searching works both ways: The user may start from two descriptors and look for all documents that can be reached from **both** descriptors, or she may start from two documents and look for descriptors that can be reached from **both** of these objects. (Descriptors that two relevant documents have in common are likely to be good descriptors for further search.) Moreover, as will become clear below, the same principles apply for searching in a database system that deals with any type of objects, such as food products, and that uses relationships or links to store data about these objects.

Stressing commonalities is not to ignore or trivialize the very real differences between existing systems and how they are searched. There are indeed many ways for finding the next item of information and for organizing the entire pattern of search. We need a systematic analysis of these many ways of searching. Figure 7 offers a multidimensional design space as the basis for such an analysis.

From a user's point of view the best system is one that supports any type of search equally well, no matter where it falls in this multidimensional design space. As a practical matter, most systems support some types of searches and some search features better than others; in that sense, the dimensions discussed can be considered as dimensions for the classification of **systems**.

Figure 7. **Dimensions for analyzing searches and systems** (simplified)

Dimension	**Typical hypermedia search**	**Typical bibliographic search without interaction**
A Type of starting object(s), their role, and method of finding them	A paragraph, a picture, an audiovisual object Mostly objects of value in their own right Found through natural encounter during a search. Object currently examined as default starting object	A search key: A subject descriptor, person, organization, etc. Mostly objects used only for searching
B Method of specifying starting object(s) and link types	Select by clicking on object and link type displayed on the screen	Deliberate selection Entering elements from the keyboard
C Type of target objects found. Granularity	Full text, picture, etc. Individual paragraphs, images, or sound objects	References to documents Whole documents
D Nature of the search interaction	Many search steps, each finding a few items to be fitted into a mosaic being built or used as stepping stones for further searching	One search step returning the final answer set of (often many) documents to be read
E Completeness and complexity of search specification for each search step	Partial, often implicit Simple	Complete, often carefully worked out Complex

3 Elements of a unified information structure

An **information base** (defined to include database, knowledge base, bibliographic database, and hypermedia base) is made up of **objects** (also called **entities** or **nodes**), **relationships** (also called **links**), **neighborhoods** (also called **regions**, **clusters**, or **virtual composites**), **queries**, and **paths and scripts** (see Figure 8 for definitions). The data base literature uses the terms entity and relationship; the hypermedia literature uses object and link.

Objects are linked through relationships into **statements** that make assertions about objects and thus carry the information in the information base. Object types and relationship types together specify what kind of statements can be made, what kind of data can be expressed in an information base; they define the **conceptual schema** of the information base.

Figure 8. **Elements of information structure**

Objects (entities, nodes) (Section 3.1)

Relationships (links) (Section 3.2)

Neighborhoods (Section 3.3)

A neighborhood is any group of objects, particularly a group identified through relationships or links with one or more other objects or neighborhoods, together with the relationships or links that exist among the members of the group. A search results in a neighborhood.

Queries (Sections 4, 9)

A **query** specifies a search and thus a neighborhood; that specification is dynamic: the neighborhood contains the objects meeting the search criteria at the time the query is invoked.

Paths and scripts (Section 8.3)

A path is a special type of object or node that defines a neighborhood of objects and specifies a sequence of these objects. (A path leads the user along a given sequence of paragraphs and figures in a hypermedia base, thus emulating a traditional document.) A script is a special type of object consisting of instructions that orchestrate the display of other objects to a user.

Neighborhoods, queries, paths and scripts, and relationships can be treated as objects.

3.1 **Objects (entities, nodes)**

A standard hypermedia system deals with media objects or documents in the broadest sense – paragraphs, pictures, sound objects – without differentiation of object types, and many systems do not differentiate between link types either. However, "To the user, nodes and links are filled with meaningful contents and organized into meaningful structures" (Halasz 1988), and systems should model and exploit this semantic richness. A typical database covers objects such as persons, organizations, food products, or technical products and uses distinct relationship types to express data about them. A generalized database/hypermedia base structure can cover a wide variety of object types and represent almost any type of information (see Figure 9 for examples).

Hypermedia systems with untyped objects and links do not represent meaning except at a very coarse level. However, the future belongs to systems that deal with meaning and can thus provide more intelligent support. For example, Carlson's 1990 system deals with strategies, objectives, and issues and their causes. It allows for sophisticated retrieval and processing through its rich semantics of object types and link types. While on the surface each object in this system may look just like a text object, object types are distinguished by what the text represents. A hypermedia system in the area of mathematics might include assertions (mathematical theorems) as objects so that they can be linked to other objects, such as persons or proofs.

Figure 9. **Some object types**

Document / Media object (text, graphics, sound) of any size	Person
Path	Group of persons
Database, data set	Organization
Assertion	Research/action agenda
Problem	Research/action area
Strategy	Project (research project, action project)
Objective	Computer program
Issue	PhysicalObject, including the following
Situation, circumstance (which may be the cause for an issue)	Organism
Term	Food product
Concept (subject, topic)	Building
	Work of art
	Technical product, device (anything from a screw to an engine to an entire airplane)

3.2 **Relationships (links) and connections**

An information base consists of stored objects (or references to objects that exist outside) and statements about these objects. The statements consist of a relationship with one or more arguments, with each argument slot filled by a specific object value; for example,

Document D8	<*supports*>	Document D2 (a statement about two documents), or
Brownie	<*hasIngredient*>	Chocolate (a statement about two food products).
Person	<*hasInterest*>	[Concept, Intensity]

A relationship that has two arguments is called **binary**; it establishes a **link** between its two arguments. Hypermedia systems use mostly binary relationships called links. Database and expert systems use many binary relationships but also relationships with three or more arguments. We use the specific term *link* when the emphasis is on binary relationships, particularly in the context of using these relationships for navigation in a hypermedia system, and the more general term *relationship* whenever the context requires it.

Figure 10 gives examples of relationship types; they illustrate the wide range of information that can be represented in a unified information structure. The relationships are usually given in just one direction, but all are bi-directional; for example, <*produces*> / <*producedBy*>.

Some relationships can be applied to many types of objects. For example, <*producedBy*> can link any type of object, such as a document, computer program, or food product to a person, organization, machine, or even an event. (In specific circumstances, one might use a more specific relationship type, such as <*authoredBy*>, <*publishedBy*>.) <*contradictedBy*> could link a mathematical theorem to a counterexample or the campaign statement of a politician to a set of facts. <*continuedBy*> is a navigational link that applies primarily to documents but also to the sequence of elements in a classification or of courses in a meal; [Object-1, Path] <*continuedBy*> Object-2, has three arguments to express the fact that an object can participate in many paths and the continuation object depends on the path. Finally, the relationship <*hasNarrowerTerm*> illustrates that thesaurus relationships are no different from any other relationships in the system, such as <*covers*> (or <*dealsWith*>), and can be used the same way in searching.

Other relationships are specific to an object type, such as documents and data sets: <*hasSummary*> leads from a long to a short form (a reader looking at the summary but requiring more detail would pursue such a link in the other direction). <*hasPrerequisite*> helps a user who does not understand a document to acquire the requisite knowledge; it also supports constructing a didactically sound path. <*hasSameContentAs*> serves to select from several documents that say the same thing the one that best fits the user's language ability and cognitive style, to warn a user who is about to read something that merely repeats what she just read, or, conversely, to suggest a different presentation if the user has trouble understanding.

Figure 10. **Some relationship types (link types)** (types defined based on meaning)

General relationship types

<producedBy>
<authoredBy>
<issuedBy>

<hasTargetAudience>

<supportedBy>

<contradictedBy>

<praisedBy>

<criticizedBy>

Object	*<covers>* *<dealsWith>*)	Concept
Inverse:		
Concept	*<coveredIn>*	Object
Term	*<occursIn>*	Document
	<describes>	
	<includes>	
	<hasSpecialCase>	
Term	*<designates>*	Concept
Concept	*<hasNarrowerTerm>* (*<NT>*; *<BT>*, *<RT>*,..)	Concept
[Object, Path]	*<continuedBy>*	Object

Relationships on issues, objectives, strategies

Document	*<helpfulFor>*	Problem
Inverse:		
Problem	*<helpIn>*	Document
Circumstance	*<causes>*	Issue
Objective	*<addresses>*	Issue
Objective	*<addresses>*	Cause
Strategy	*<aimsAt>*	Objective
Strategy	*<assignedTo>*	Organization

Relationship types primarily for documents

Document	*<hasPrerequisite>*	Document
Document	*<hasSummary>*	Document
Document	*<hasSameContentsAs>*	Document
Document	*<isSimplifiedFrom>*	Document
Document	*<isLaterVersionOf>*	Document
Document	*<writtenIn>*	Language
Document	*<cites>*	Document
Document	*<illustrates>*	Entity

Relationship types for organizations and projects

Person	*<affiliatedWith>*	Organization
Person	*<directs>*	Project
Organization	*<funds>*	Project
Organization	*<carriesOut>*	Project

Relationships on food products

FoodProduct *<hasIngredient>*
[FoodProduct, Rank, Total %, Solids %, [Purpose list]]

FoodProduct *<underwentProcess>*
[Process, Equipment, Temperature, Duration, Place/stage, SequenceNo., [Purpose list]]

Food product *<hasConstituent>*
[ChemicalSubstance, Rank, Total %, Solids %]

Relationships for user model

Person	*<hasInterest>*	[Topic, Intensity]
Person	*<hasKnowledgeOf>*	[Topic, Depth]
Person	*<readsLanguage>*	[Language, Fluency]
Document	*<readableBy>*	[object1, object2]
Document	*<processableBy>*	object

<*hasIngredients*> must be able to model actual data about a food product in all their complexity: Brownie <*hasIngredient*> Chocolate does not tell the whole story; one also wants to know that chocolate has rank three in order of predominance, amounts to 20% based on total weight, contributes 25% of the food solids (the food components after all moisture is removed), and has the purposes of improving taste and achieving chewy consistency. Modeling this complexity requires a relationship with six arguments:

> Brownie <*hasIngredient*> [Chocolate, rank 3, percent of total weight 20%, percent of solids 25%, [purposes: improving taste, chewy consistency]].

To appreciate the need for this complexity, consider the ingredients of a custard: If it is made with fluid milk, milk is listed as the first ingredient by total weight, but if it is made with dry milk, milk is listed as the third or fourth ingredient, even though the content of milk solids is the same, as could be seen from full ingredient information. Statements formed with the relationship *has-ingredient* serve many functions: they inform the user about the ingredients of a food, they let the user find all foods containing a given ingredient, such as chocolate chips or chocolate in any form, they let the user find all ingredients that have been used to achieve chewy consistency, etc.

An information system should also include **user models**, data about users (persons, organizations, machines), their characteristics and needs. User model data are often kept separately, but they can and should be treated just like any other data, represented as statements in the database. Much of the time, data that tell us something about the user are useful for other purposes as well; for example, data about the projects a user is working on guide the information provision to that user but are also used for project management. Document <*readableBy*> (Object1, Object2) refers to the physical readability of a document by object1 (a person or a computer) with the aid of object2 (another device, such as a microform reader or a disk drive). <*processableBy*> refers to ability of the agent object to do something with the data read, e.g., a file or document being processable by a computer program or a text being processable by a person (I can physically "read" a printed Swahili text, but I cannot process it).

The entity types in Figure 10 should be looked at as important examples, not as restrictions. For example, <*hasPrerequisite*> also links computer programs A and B where running A requires B. <*writtenIn*> applies to both documents and computer programs. <*hasIngredient*> applies also to drugs.

Sometimes one is interested in objects more than one step away from a starting point, such as starting from an organization and finding documents authored by any of its members. Then one would follow a chain of links or **connection** (discussed in detail in Section 8.2):

> Organization <*hasMember*> Person, Person <*isAuthorOf*> Document

To find the business phone number of a person, follow the connection

> Person <*affiliatedWith*> Organization, Organization <*hasPhone*> PhoneNumber

Named connections express expert knowledge; they define patterns of inference for use in retrieval. Named connections are a notation for writing simple Prolog rules.

Figure 10. **Some relationship types (link types)** (types defined based on meaning) repeated

General relationship types

<*producedBy*>
<*authoredBy*>
<*issuedBy*>

<*hasTargetAudience*>

<*supportedBy*>

<*contradictedBy*>

<*praisedBy*>

<*criticizedBy*>

Object	<*covers*> <*dealsWith*>)	Concept
Inverse:		
Concept	<*coveredIn*>	Object
Term	<*occursIn*>	Document
	<*describes*>	
	<*includes*>	
	<*hasSpecialCase*>	
Term	<*designates*>	Concept
Concept	<*hasNarrowerTerm*> (NT; BT, RT,..)	Concept
[Object, Path]	<*continuedBy*>	Object

Relationships on issues, objectives, strategies

Document	<*helpfulFor*>	Problem
Inverse:		
Problem	<*helpIn*>	Document
Circumstance	<*causes*>	Issue
Objective	<*addresses*>	Issue
Objective	<*addresses*>	Cause
Strategy	<*aimsAt*>	Objective
Strategy	<*assignedTo*>	Organization

Relationship types primarily for documents

Document	<*hasPrerequisite*>	Document
Document	<*hasSummary*>	Document
Document	<*hasSameContentsAs*>	Document
Document	<*isSimplifiedFrom*>	Document
Document	<*isLaterVersionOf*>	Document
Document	<*writtenIn*>	Language
Document	<*cites*>	Document
Document	<*illustrates*>	Entity

Relationship types for organizations and projects

Person	<*affiliatedWith*>	Organization
Perso	<*directs*>	Project
Organization	<*funds*>	Project
Organization	<*carriesOut*>	Project

Relationships on food products

FoodProduct <*hasIngredient*>
[FoodProduct, Rank, Total %, Solids %, [Purpose list]]

FoodProduct <*underwentProcess*>
[Process, Equipment, Temperature, Duration, Place/stage, SequenceNo., [Purpose list]]

Food product <*hasConstituent*>
[ChemicalSubstance, Rank, Total %, Solids %]

Relationships for user model

Person	<*hasInterest*>	[Topic, Intensity]
Person	<*hasKnowledgeOf*>	[Topic, Depth]
Person	<*readsLanguage*>	[Language, Fluency]
Document	<*readableBy*>	[object1, object2]
Document	<*processableBy*>	object

The variety of relationship types reinforces the unified system theme. On the surface, a statement about the ingredient of a food product may look quite different from a link between two documents, such as document A <*criticizes*> document B. But they can be expressed in exactly the same format, and there are many advantages in doing so. The information structure management model combines the selective and inferential power of database and expert systems with the interface and navigational power of hypermedia systems. One integrated conceptual schema is parsimonious: Many relationship types apply across content areas. More importantly, an integrated schema allows for useful relationships that would be much more cumbersome in separate systems. For example, a user could find all food products with 20% or more chocolate, select one of them, follow a link to a text describing it or a picture depicting it; or he could define the neighborhood of all documents that describe the selected product or any of its ingredients. Or a user could apply a program that computes the nutrient content data of a prepared food product based on data on the ingredients and nutrient data for each the ingredients, and then pick a nutrient on the list and retrieve all documents that discuss the importance of this nutrient.

3.3 **Neighborhoods and queries**

Simple objects are not enough to model the complexities of the real world, for example the composition of documents from multiple parts. We therefore introduce the concept of "neighborhood" which will take on particular significance in enabling powerful searches as discussed in Section 4. A neighborhood (also called region or cluster) is any set or group of objects together with the relationships that exist among them (Figure 11). A neighborhood can in turn contain neighborhoods. The term was chosen in keeping with the spatial and navigation metaphor often associated with hypermedia systems.

Usually the members of a neighborhood are selected based on their relationship with one or more other objects. In the simplest case, a neighborhood consists of all objects that can be reached from a starting object through links of a given type – the neighbors of the starting object. An example is the neighborhood consisting of all documents in which a given document is criticized; this neighborhood is assembled by starting from the given documents and following links of the type <*criticizedBy*>. There is a direct association between neighborhood and search: A query leads to a neighborhood. Including queries as objects allows for any kind of statements one wishes to make about a query and thus about the query-defined neighborhood as a whole.

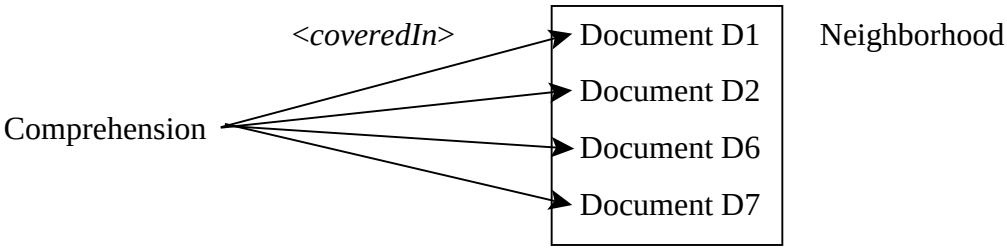
So far we have discussed the most common way of defining a neighborhood, definition through a query. A neighborhood can also be defined by enumerating its elements. For example, a user may mark objects encountered in the course of a search for later examination and processing. The search might be for objects to be included in Exhibit-25 on *apocalypse in art*; the user does a broad search and marks the objects worthy of consideration for the exhibit. In doing so, the user in effect updates the information base: He adds an object that stands for the neighborhood being defined, namely Exhibit-25, and <*includes*> links from that object to all the members of the neighborhood, such as Exhibit-25 <*includes*> Art-object-337. Thus in the future the elements of neighborhood can be found by the query Exhibit-25 <*includes*> X.

Figure 11. **Neighborhoods**

Neighborhood: any group of objects or neighborhoods, particularly a group identified through links with one or more other objects, together with the relationships that exist among them.

Examples

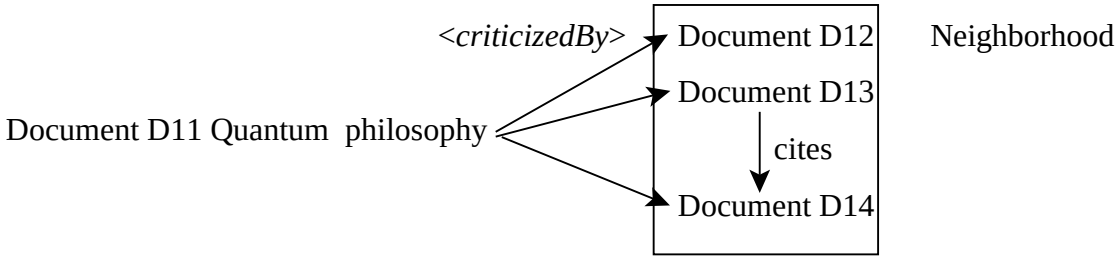
All documents in which a given topic, such as *comprehension* is *covered*



All objects <*producedBy*> a given person, e.g. all music composed by Gershwin

Search: Gershwin *produces* object

All documents in which a given document D1 is *criticized*, together with the relationships among them (such as one of these documents citing another)



All pictures *illustrating* a given paragraph

All persons *producing* a given document

All food products *having-ingredient* a given food product

All food products that occur in a <*hasIngredients*> statement in the ingredient role associated with the purpose chewy consistency

All objects selected by the user during the course of a search for later examination and processing

3.3.1 **Offspring neighborhood. Document as a tree of smaller and smaller units**

Neighborhoods based on hierarchical relationships between objects are very important for searching, as discussed more fully in Section 4.6. Following a hierarchical relationship downward yields an **offspring neighborhood**; see Figure 12 for examples.

An object that heads an offspring neighborhood plays an important role as **organizing node**, establishing a "composition mechanism" as "a way of representing and dealing with groups of nodes and links as unique entities, separate from their components" (Halasz 1988, p. 843). Links associated with an organizing node are either non-inheriting or inheriting.

A **non-inheriting** link pertains to the organizing node as such or to the totality of the nodes under it; for example, Scientific American <*covers*> All of science is a non-inheriting link since it applies only to the journal (the totality of articles) as a whole, not to individual articles.

An **inheriting link** applies to every object in an offspring neighborhood formed along a given relationship; for example Scientific American <*hasTargetAudience*> Educated lay person is an inheriting link since it applies to every article <*includedIn*> Scientific American. Inheriting links are introduced as a space-saving device: It is more efficient to assign <*hasTargetAudience*> once to Scientific American as an inheriting link than to assign it to every single article.

Figure 12. **Offspring neighborhoods. Examples**

A book and its chapters (one level down)

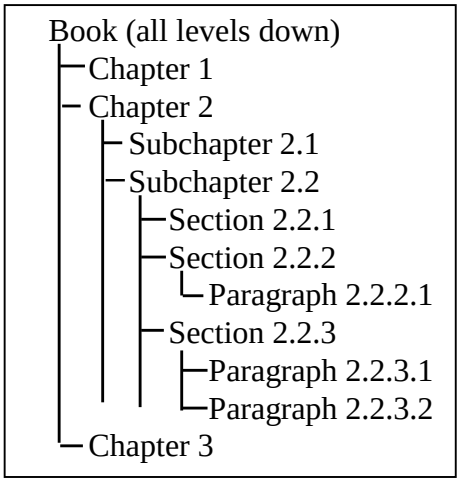
A book and its chapters, sub-chapters, sections, and paragraphs (All levels down)

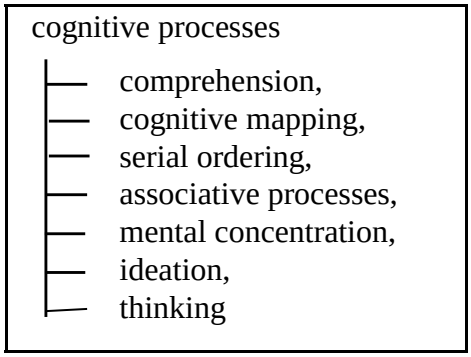
A website and all its sub-websites, pages, and parts of pages as identified by anchors

A journal and its volumes
A journal volume and its issues
A journal issue and its articles.

A concept and its narrower concepts

An assertion and its special cases





3.4 Links to, from, and between neighborhoods

Neighborhood links are essential in searching; they are defined in terms of links between atomic objects as follows (Figure 13):

A **neighborhood outlink** from neighborhood N1 is any link **from** any object in N1 to some atomic object.

Example 1: **A citation from a journal** is defined as a citation from any article in the journal.

Example 2: Consider **a search for all documents in which document D is criticized**; such documents can be reached from document D following <*criticizedBy*> links. The search should also find documents criticizing **any part of D**; thus it should start from the offspring neighborhood arising from the top node of D (see Fig. 12) and follow the relationship <*criticizedBy*> from any object in that offspring neighborhood.

A **neighborhood inlink** to neighborhood N2 is any link from an atomic object that ends up in an object in N2.

A **neighborhood-to-neighborhood outlink** from N1 to N2 is any neighborhood outlink **from** neighborhood N1 **to** neighborhood N2, that is, an neighborhood inlink to N2.
A **neighborhood-to-neighborhood inlink** is the reverse.

Example 1: A journal-to-journal citation outlink exists whenever any article in journal J1 cites any article in journal J2.

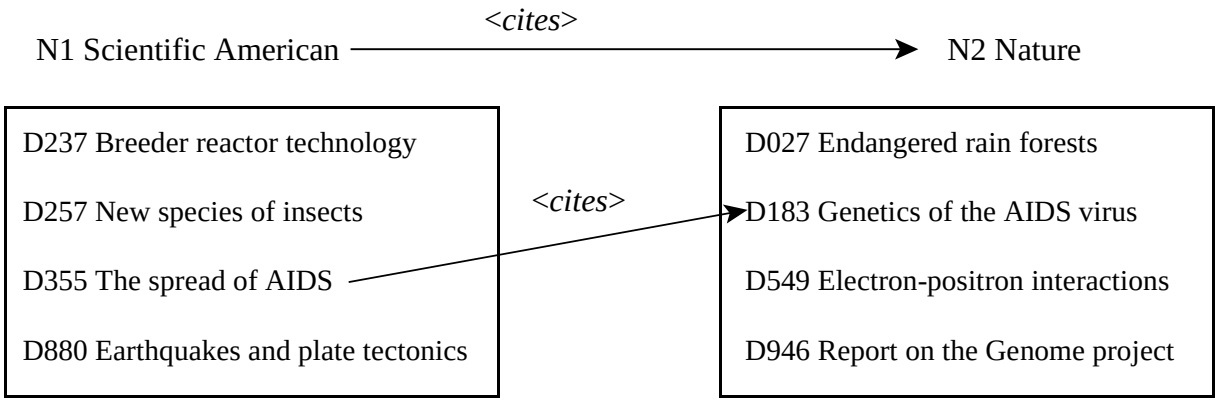
Example 2: A website-to-website link exists whenever any page in website S1 links to any page in website S2. The link structure of the Web can be analyzed in terms of page-to-page links or in terms of site-to-site links.

Figure 13. **Neighborhood links**

Neighborhood outlink (from N1): A link **from** any object that is in neighborhood N1

Neighborhood inlink (to N2) : A link **to** any object that is in neighborhood N2

Neighborhood-to-neighborhood link: A link **from** any object in N1 **to** any object in N2



D355 The spread of aids <*cites*> D183 Genetics of the AIDS virus

can be interpreted in several ways:

as a outlink from D355 to D183 (an inlink of D183);

considering that D355 is in N1, as a neighborhood outlink from N1 to D183;

considering further that D183 is in N2, as a neighborhood-to-neighborhood outlink from N1 to N2;

and conversely as a neighborhood-to-neighborhood inlink to N2 from N1.

Weighted neighborhood links are sometimes useful: Instead of requiring a link from just one element of N1, require a link from two (or n); let's call this a neighborhood-n-outlink.

Example: given a neighborhood N1 of documents (such as a search result), find all terms that are the target of an outlink from at least two documents in N1. (Terms that are the target of such a neighborhood-2-outlink from two document neighborhoods N1 and N2 are bridge terms between N1 and N2, Swanson and Smalheiser 1997).

Some systems allow the user who has done a search resulting in a neighborhood N1 of documents to list terms in descending order of frequency of occurrence in N1. In the model discussed here, the terms are targets of weighted neighborhood-n-outlinks from N1 sorted by the weight n.

****

This completes the discussion of the basic elements of an information structure. Further elements that make for a richer structure to express still more types of information or knowledge and do more powerful searches are discussed in Section 8.

The next section applies the structure we defined to the analysis of many different types of search, starting with the simple and moving to the quite complex..

4 Search

This section discusses the application of the information structure discussed to its ultimate purpose: sophisticated and powerful searching. Searching can be a very complex task, but much of this complexity can be hidden from the user who is willing to accept choices made by the system. Complexity behind the scenes adds power and makes the user's life easier. Alternatively, a search wizard can guide the user through the process of designing his own search, showing all options.

4.1 Definition of search

A total search consists of one or more **search steps** as illustrated in Figure 14.

Figure 14. **Sample searches**

Example 1. Navigation search

Step 1: A user begins with a **problem**. The <*helpIn*> link leads to helpful documents.

Step 2: He starts from one of these documents and uses the <*hasPrerequisite*> link to find documents she needs in order to acquire required background knowledge.

Step 3: He selects one of the prerequisite documents, finds it too difficult, and uses it as a new starting point; the <*hasSimplifiedForm*> link leads him to an easier document.

Example 2. Navigation search

Step 1: A user starts with a term and, using the <*designates*> link, finds the corresponding concept(s), selects the meaning he intends, finds the preferred term for the intended meaning, as well as broader and narrower concepts (a concept neighborhood).

Step 2: She starts from this concept neighborhood and, using the <*coveredIn*> link, finds documents about or relevant for the concept.

Example 3. Search based on intrinsic structure

A user starts with two words and looks for all documents in which the two words occur adjacent to each other, for example, *information* ADJ *retrieval*. This search does not use navigation along links. Rather it requires the system to look **inside** documents to find the ones that meet the requirement.

Example 4. Similarity search

A user starts with a known relevant object and looks for objects that are similar in terms of their links to other objects or in terms of their internal characteristics or a combination thereof. The user must specify the criteria for judging similarity. For example, the user may look for other objects with similar descriptor assignments or with similar producers/authors or for documents containing similar words or for a piece of music containing sequences of notes similar in structure to those found in the known piece. The system must be able to compute similarity.

This section discusses searching based on the **navigation metaphor**: following links from a known starting point to an unknown target. Each search step starts from something that is known, often a **starting object or neighborhood**, and leads to one or more **target objects or neighborhoods** that, one hopes, contribute to the goal. Each object or neighborhood found along the way may contain some (or all) of the information needed and/or may serve as a stepping stone to further information, as the starting point in the next search step.

Figure 15. **Search: Definition**

Total search = series of search steps, each leading

from something known (a known object or neighborhood)

to something unknown but expected to be helpful (one or more helpful target objects or neighborhoods)

An object or neighborhood encountered may be helpful because it

contains some of the information needed and/or

serves as a stepping stone to find other objects or neighborhoods.

Examples 3 and 4 illustrate different kinds of searches; these searches require a query formulation. Navigation-based searches could also be expressed as queries; query-based searching covers all kinds of searches, including searches based on characteristics that are intrinsic to the target objects rather than relying solely on relationships between objects, similarity searches, and searches that extract data from databases using a query formulation language such as SQL. A complete classification of types of searches is presented in Section 9.

In any kind of search the result may be more differentiated than just giving a set of objects judged relevant by the system and thus retrieved (while rejecting altogether objects not considered relevant). The reality of relevance, or what would be helpful to the user, is more complex; relevance is a matter of degree. All types of searches can be adapted to return a list of target objects that is ranked by a criterion that is related in some way to expected relevance.

4.2 Specification of a search based on relationships

This section discusses in detail how to specify a search based on using relationships from one or more starting objects to identify target objects. For ease of explanation the discussion uses the navigation metaphor, but the concepts apply also to query-based search using relationships.

In each search step the user must tell the system what he needs through a **search specification (query formulation)** consisting of the four elements shown in Figure 16.

Figure 16. **Elements of a search specification (query formulation) for a single search step**

The general type of target objects or neighborhoods

One or more search criteria, each consisting of

A starting object or neighborhood

A connection condition: the permissible connections from the starting object or neighborhood to the target objects or neighborhoods.

The display format for data about the target objects or neighborhoods found (or the objects themselves)

The **target object or neighborhood specification** expresses what the user wants to find – concepts, persons, documents, offspring neighborhoods of documents, ancestor neighborhoods of documents, assertions, etc. Specifying neighborhoods as targets is significant in that it determines the scope of a Boolean AND search as explained below.

The first component of a **search criterion** is a **starting object or neighborhood**. For the first search step the user may employ either an object she knows (a problem, a document, a term, a person) or she may select an object from an initial menu. Or the user may enter a **starting object type**, such as **problem**, and in return be shown a menu of possible values to select from. For the search steps after the first, the starting objects are what the first step has found.

The first component of a **search criterion** is a **connection condition** that specifies how to get from the starting object to the target objects wanted. A connection condition consists of **one or more permissible connection types**. Often the user's purpose is achieved best by allowing any of a number of related connection types "in parallel" to get from the starting point to the targets; in the example that follows, several connection types to be used in parallel are shown in {}. Assume a user is looking at a given document D1 and wants to find documents that criticize D1 to find out whether she can trust D1. The user should use several connection types in parallel; the examples illustrate connection types that lead to relevant documents.

Connection 1: [Document D1 <*commentedBy*> Document D8],
Connection 2: [Document D1 <*criticizedBy*> Document D9],
Connection 4: [Document D1 <*proposes*> Theory T,
Theory T <*commentedBy*> Document D10],
Connection 5: [Document D1 <*proposes*> Theory T,
Theory T <*criticizedBy*> Document D20]}.

Each of the connections in this "bundle" will lead to relevant or potentially relevant documents.

The **display format** indicates what the user wants to see for each object or neighborhood found. For example, for text documents the user may want to see just titles, or the author (possibly with organizational affiliation) and the title, or the full text. For images, the user may want to see a thumbnail and copyright and license fee information, or she may want picture files downloaded in JPEG format. The system gathers this information in an implied search (transparent to the user). The format also specifies the arrangement of the objects or neighborhoods found (by date, by author, by degree of expected relevance, as a network based on a specified link type, etc.).

The following sections discuss single-step navigation searches; we begin with the simple – single-criterion searches starting from a single object and go on to the complex – combination searches with neighborhoods as targets and searching with hierarchical inheritance. Figure 17 shows the different search types arranged by the two dimensions **search criteria** and **search targets** and gives the section in which each is discussed.

Figure 17. **Types of navigation searches**

Search criteria	Type of target	
	Single objects as targets	Neighborhoods as targets
Search using a single criterion		
starting from a single object	Section 4.3.1	Section 4.3.2
starting from a neighborhood	Section 4.4	
Search using a combination of multiple criteria (Boolean AND search or weighted search) starting from a single object or from a neighborhood	Section 4.5.1	Section 4.5.2
Hierarchical inheritance through ancestor neighborhoods as search targets Single criterion or combination		Section 4.6

4.3 **Single-criterion search starting from a single object**

In a single-criterion search, a single connection going into a target object or neighborhood is sufficient to select that object

4.3.1 **Single-criterion search starting from a single object with single objects as targets**

This is the simplest kind of search. Examples are given in Figure 18.

Figure 18. **Single-criterion search starting from a single object. Examples**

Example 1

STARTING OBJECT: A subject descriptor

TARGET OBJECTS: Documents

CONNECTION CONDITION: <*coveredIn*>

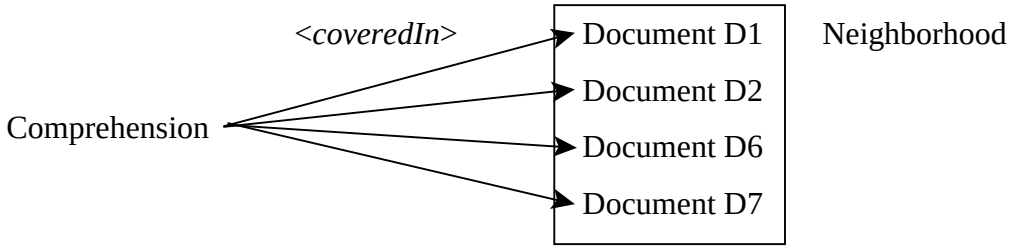


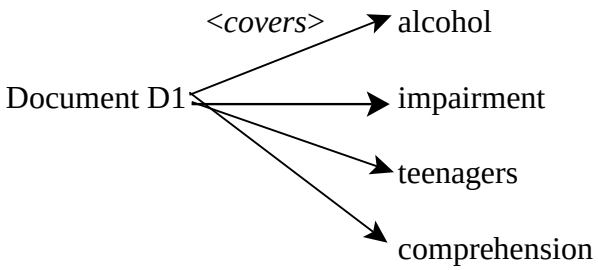
Figure 18. **Single-criterion search starting from a single object. Examples**. Continued

Example 2

STARTING OBJECT: A document
TARGET OBJECTS: Descriptors

CONNECTION CONDITION: <*covers*>

(The descriptors found might be useful as starting points in further search steps)



Example 3

STARTING OBJECT: A building

TARGET OBJECTS: Documents

CONNECTION CONDITION: {<*depictedIn*>, <*coveredIn*>}

Example 4

STARTING OBJECT: A book (only the top node for the book as a whole, excluding the nodes for parts of the book)

TARGET OBJECTS: All objects of any type

CONNECTION CONDITION: *Universal link* (any link type)

Example 5

STARTING OBJECT: A book (top node only)

TARGET OBJECTS: All objects of any type

CONNECTION CONDITION: Any one-link or two-link connection

4.3.2 **Single-criterion search starting from a single object with neighborhoods as targets**

In the examples in Figure 18 (previous page) single objects are specified as targets for selection. Sometimes one may want to retrieve whole neighborhoods, for example, whole documents (such as journal articles or books) rather than single paragraphs (Figure 19, on facing page). Put differently, the whole document should be shown if any of its subordinate objects (sections, paragraphs) are found. In that case, the search targets can be specified as document offspring neighborhoods. The system finds any document that fulfills the search criterion and then identifies the whole document to which it belongs. In this example, specifying neighborhoods as targets of a single-criterion search does not affect retrieval per se but what information is displayed once an object is found. This is in contrast to combination searches where, as we shall see, it makes a big difference for retrieval whether atomic objects or neighborhoods are specified as targets.

The second example (Figure 20, below on this page) is more interesting. The type of target neighborhood is a food product with all its ingredients and their constituents, enabling an allergy sufferer to find all offending foods.

Figure 20 out of order due to layout constraints

Figure 20. **Single-criterion search starting from a single object with neighborhoods as targets. Example 2**

A search for all food products to which a person (or person class) is allergic. This search is complex because <*allergicTo*> statements can refer to food products, such as milk, or to a chemical substance, such as lactose, and because the offending allergen may be in any ingredient of the food. In other words, this search must find all food products that directly or indirectly contain something the person is allergic to.

STARTING OBJECT: A person

TARGETS: Neighborhoods consisting of a food product, all food products reachable from it through <*hasIngredient*> links or chains of <*hasIngredient*> links, and all chemical substances reachable from any of these food products through <*hasConstituent*> links

CONNECTION CONDITION: <*allergicTo*>

An <*allergicTo*> link from the person into any element of a target neighborhood is sufficient.

DISPLAY: Show the whole food and the offending ingredients or constituents.

Figure 19. **Single-criterion search starting from a single object with neighborhoods as targets**

Search specification

A search for whole documents that deal with a given descriptor, such as **Drug treatment**. The descriptor could be assigned to the document as a whole or to any section.

STARTING OBJECT: Subject descriptor **Drug treatment**

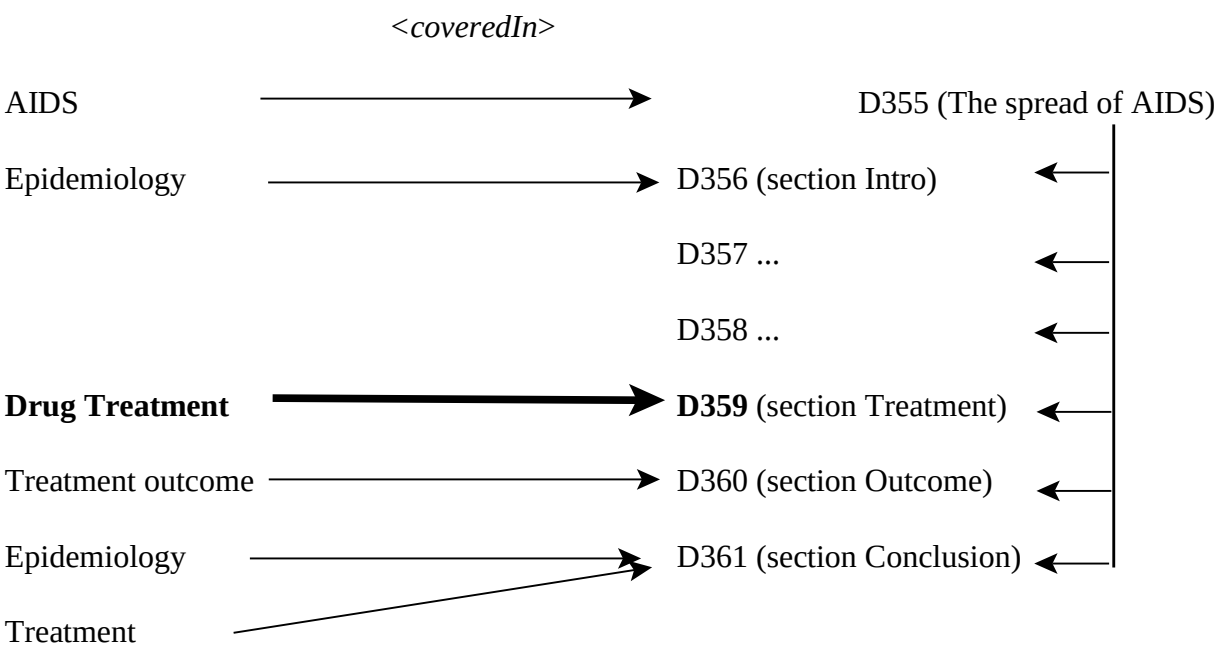
TARGETS: Offspring neighborhoods from whole document nodes

CONNECTION CONDITION: <*coveredIn*>

The connection condition is fulfilled if there is a <*coveredIn*> link from the descriptor to any one element (section, paragraph) of a target neighborhood.

DISPLAY: Show the whole document node with all its subordinate nodes. Highlight the objects to which the descriptor is assigned.

Search example



4.4 **Single-criterion search starting from a neighborhood**.

A single-criterion search starting from a neighborhood considers any object in the neighborhood as a starting point; it is an implied OR search which leads to many more objects than a search starting from a single object.

The first example (Figure 21) is an inclusive subject search: The system first builds the offspring neighborhood consisting of a descriptor and all its narrower descriptors and then starts from any descriptor in that neighborhood (rather than be limited to the one descriptor that heads that part of the hierarchy) to find documents via the connection type <*coveredIn*>. This is an implied ORing of the starting descriptors.

The second example (Figure 22) illustrates the usefulness of this method even better. An assertion is contradicted when any of its special cases is contradicted. So to see all objects that contradict an assertion, start from the assertion itself and find all objects linked via <*contradictedBy*>, but then also start from each of the special cases and follow the same link, that is, start from the offspring neighborhood.

The third example shows how to find all the documents linked to a given book: Start from the book node itself but also from all the nodes dependent on the book node via a chain of <*includes*> links, that is, an offspring neighborhood.

The fourth example defines a geographical area as a neighborhood of places and then looks for all businesses located there.

This type of search is very powerful since the user can define any starting neighborhood she wants to, using a search of arbitrary complexity. In effect, the user has the system do a preliminary search to assemble the starting neighborhood for the main search. Some systems provide shortcuts for defining certain kinds of neighborhoods. For example, when searching MEDLINE one can use EXPLODE (or, in Dialog, !) to define the offspring neighborhood of a subject descriptor (the set consisting of the subject descriptor and all descriptors under it in the hierarchy); in this case, the preliminary search is transparent to the user.

Figure 21. **Single-criterion search starting from a neighborhood. Example 1**

Any of the objects in the starting neighborhood can serve as a starting point for the search, vastly increasing retrieval (implied Boolean OR).

Start from a concept and its narrower concepts, e.g.

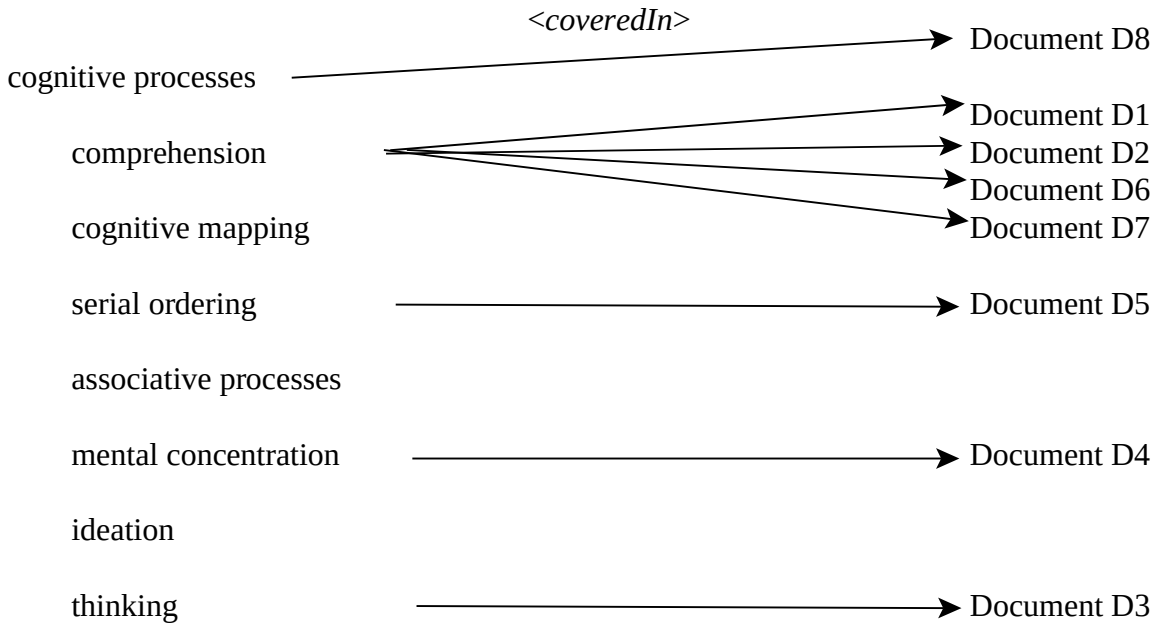


Figure 22. **Single-criterion search starting from a neighborhood. More examples**

STARTING NEIGHBORHOOD: An assertion and all its special cases
TARGET OBJECTS: All objects of any type
CONNECTION CONDITION: *<contradictedBy>*

STARTING NEIGHBORHOOD: A book and all its chapters, sections, and paragraphs.
TARGET OBJECTS: All objects of any type
CONNECTION CONDITION: The universal link

STARTING NEIGHBORHOOD: A city and all locations in a 100 mile radius
TARGET OBJECTS: All businesses
CONNECTION CRITERION: *<isLocationOf>*

4.5 Search using a combination of multiple criteria (Boolean AND search or weighted search)

Single-criterion searches are simple but often overly general. More specific selection requires using two or more search criteria simultaneously – a **combination search** (Figure 23): For a target object or neighborhood to be selected in a combination search, it must be reachable by two or more connections, normally coming from different starting objects. A combination search with direct connections, each using a single link type, is a straightforward Boolean AND. A combination search with connections of arbitrary length with arbitrary link types is called **spreading activation** in the context of semantic networks. The formalism described here lets the user specify anything in between.

4.5.1 Combination search with single objects as targets

Figure 23 gives some simple and familiar examples. The first example is a plain Boolean AND search where the searcher specifies two descriptors and wants documents that deal with both.

The second example shows the reverse: The searcher uses known relevant documents as starting points to find good candidate descriptors that can be used to find more documents in the next search step. Using the same Boolean search method, she can find all descriptors that are used in indexing **both** documents. These descriptors should be good candidate descriptors to find more relevant documents, much better than the descriptors used in indexing just a single relevant document. In the example in Figure 23, there are two known relevant documents, A and B.

Document D1 *covers* alcohol, comprehension, impairment, teenagers

Document D2 *covers* comprehension, spatial, sex differences

Clearly **comprehension** is a much more plausible descriptor for finding more relevant documents than **alcohol** or **sex differences**.

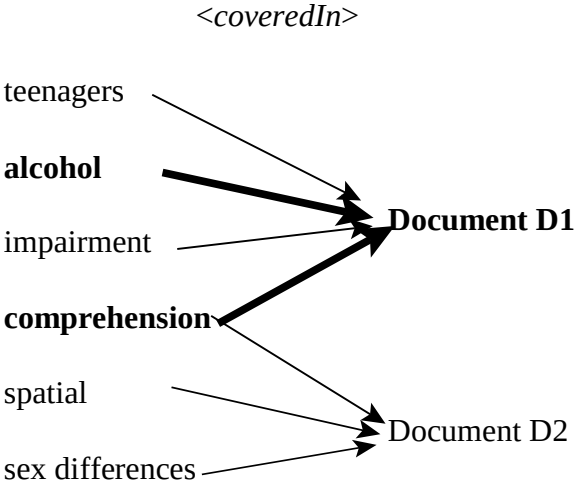
With the general search operator suggested here, this second search follows exactly the same format as the first. In an extension of this method, the searcher specifies the set of all known relevant objects as the starting neighborhood, and the system ranks descriptors by the number of objects they index. (This is the reverse of ranked retrieval of objects by starting with a set of descriptors and ranking objects by the number of descriptors covered in them, see Section 9.3.)

Figure 24 gives a further example that illustrates the need for using offspring neighborhoods as search targets, to be discussed in Section 4.5.2.

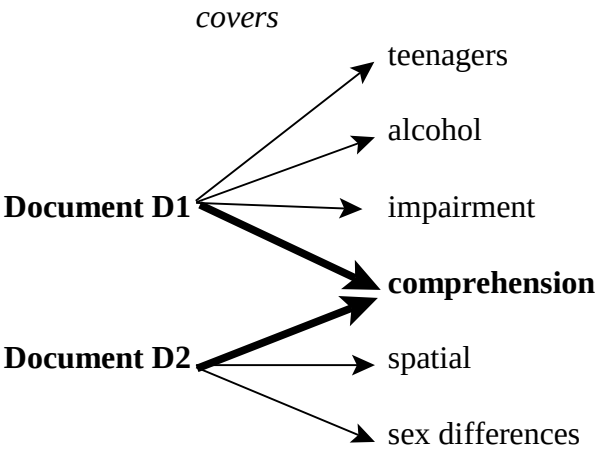
Figure 23. **Combination search (Boolean AND) with single objects as targets**

Find all objects that satisfy two or more search criteria simultaneously.

Example 1. Boolean search for documents: Starting from two descriptors, find all documents in which both descriptors are covered.



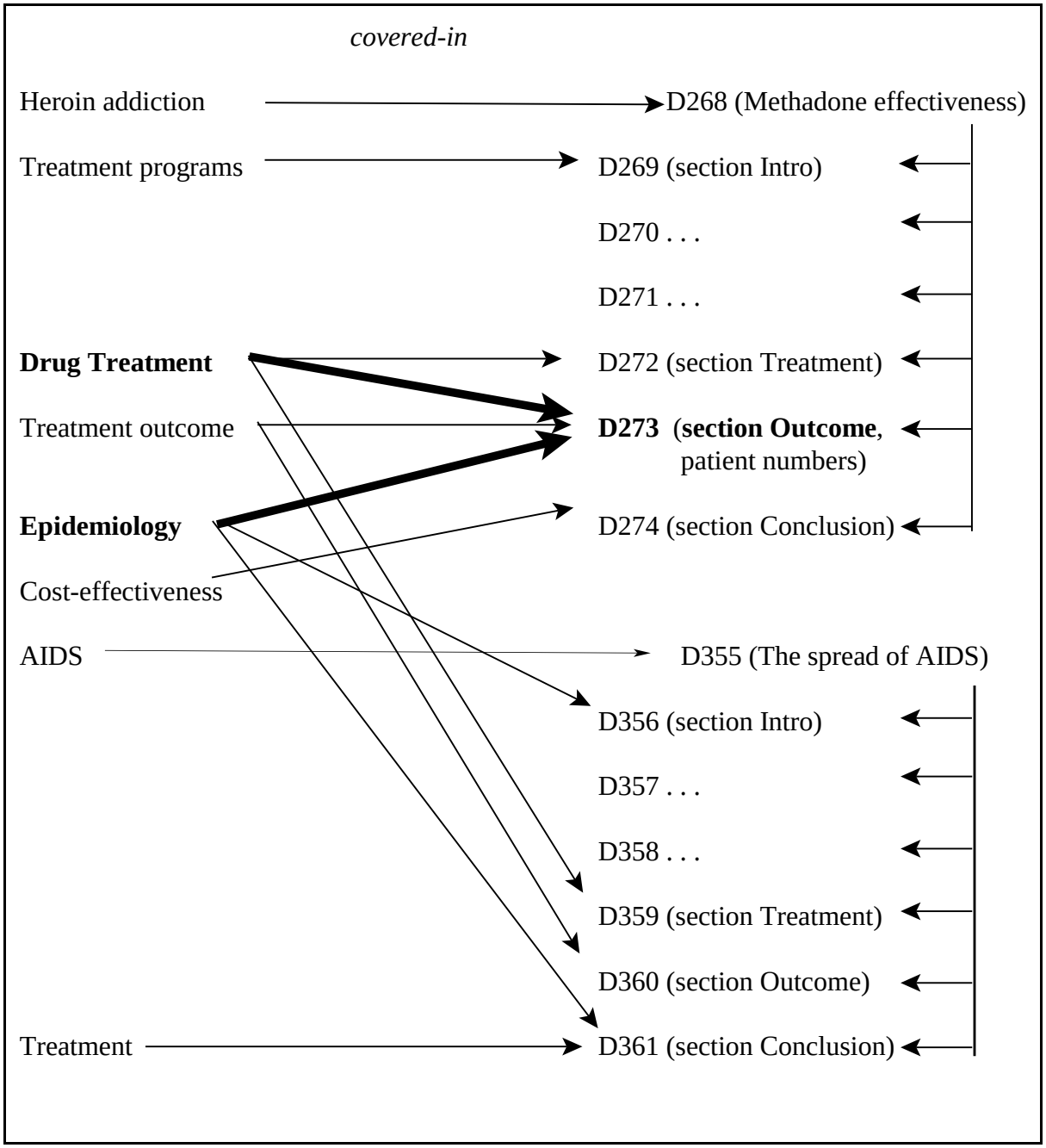
Example 2: Starting from two known relevant documents, find all descriptors that are in a <*covers*> relationship from both documents



Example 3 (cocitation; exactly the same in format as example 1):

Starting from two documents, find all documents in which both starting documents are *cited*.

Figure 24. **Combination search (AND) with individual document nodes as targets**
Illustrating the effects of a narrow scope for AND



Note that Section D273, and thus document D268, is found since it covers the two query descriptors within the same section. Document D355 is not found; the two query descriptors are covered in the document as a whole but not in a single section.

4.5.2 **Combination searches with neighborhoods as targets**

Combination search with neighborhoods as targets is among the most powerful search features. With combination searches it makes a big difference what scope we give the AND, whether we restrict the targets to atomic objects or whether we admit neighborhoods and, if so, what type of neighborhood (Figure 25). Searching for neighborhoods uses to-neighborhood connections; a to-neighborhood connection exists whenever there is a connection to any element of the neighborhood, greatly increasing the possibilities for simultaneous satisfaction of two search criteria. By proper definition of target objects or neighborhoods, the searcher can require, for example, that two terms co-occur in the same paragraph, in the same book chapter, or in an entire book, or in an entire journal issue, or in the neighborhood formed around a document and all documents it cites, or in all documents that originated from a research project, or all messages in a news group thread. The examples show that this concept has very broad application. The examples in Figures 26 and 27 further illustrate this fundamental concept..

Figure 25. **Combination search (Boolean AND) with neighborhoods as targets**

A neighborhood satisfies a search criterion if any of its objects satisfies it.

Examples:

Starting from two descriptors,
find all neighborhoods of a given type <*cover*> (are in some way relevant to) both descriptors.

Examples for neighborhood types that make sense here:

Offspring neighborhoods starting from the top node for a whole document, such as book, journal article, or report, along the link type <*includes*>.

Ancestor neighborhoods of any document along the link type <*includes*>.

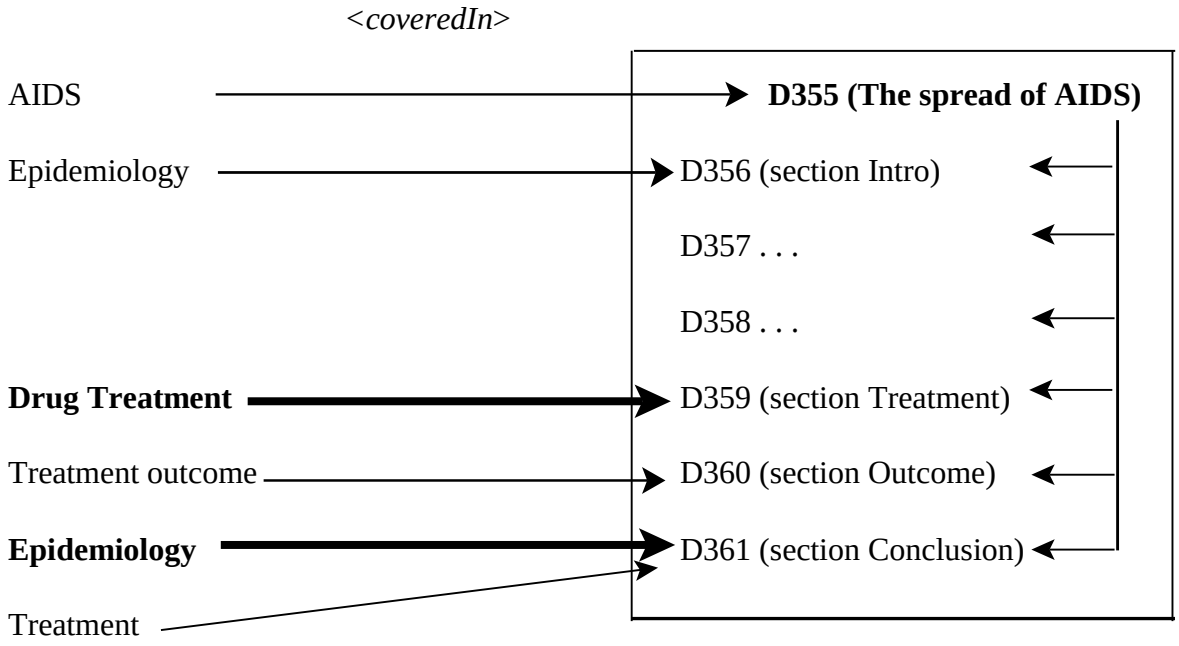
The neighborhood consisting of a document and all the documents it <*cites*>.

Example 1 (Figure 26) is straightforward: Whole documents, modeled through offspring neighborhoods, as search targets. The search specification from Figure 24 is extended to permit two descriptors to occur anywhere in a target neighborhood. Another kind of document neighborhood useful as search target would be a newsgroup thread with its various messages or a long document on the Web that is divided into Web pages (chapters) so that a regular Boolean Web search does not find the document if the two terms occur in different chapters.

Example 2 (Figure 26) generalizes the idea of finding descriptors in common to two relevant objects and using them as descriptor candidates for further searching. The search now targets **descriptor neighborhoods** that can be reached from two relevant objects. In this example, the system finds a broad descriptor in common to two documents even though the broad descriptor is represented by a different one of its narrower descriptors in each of the two documents. The broad descriptor and its narrower terms can be very useful for further search.

Figure 26. **Combination search (Boolean AND) with neighborhoods as targets.**

Example 1. Starting from two descriptors, find offspring document neighborhoods



Example 2. Starting from two documents, find offspring descriptor neighborhoods that can be reached from both through <*covers*>. Documents D2 and D3 are known to be relevant

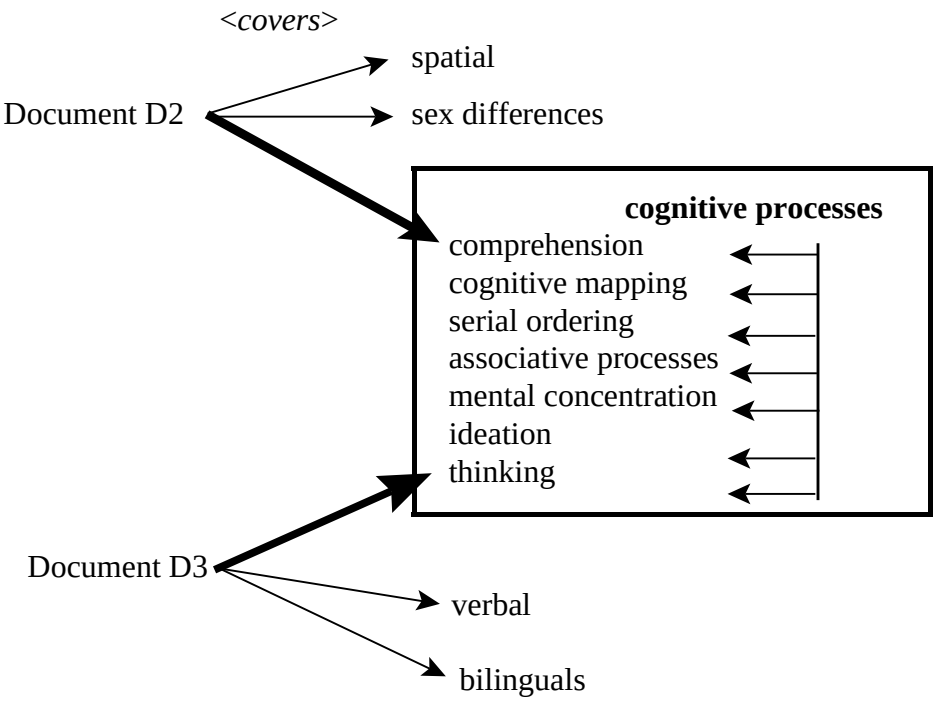
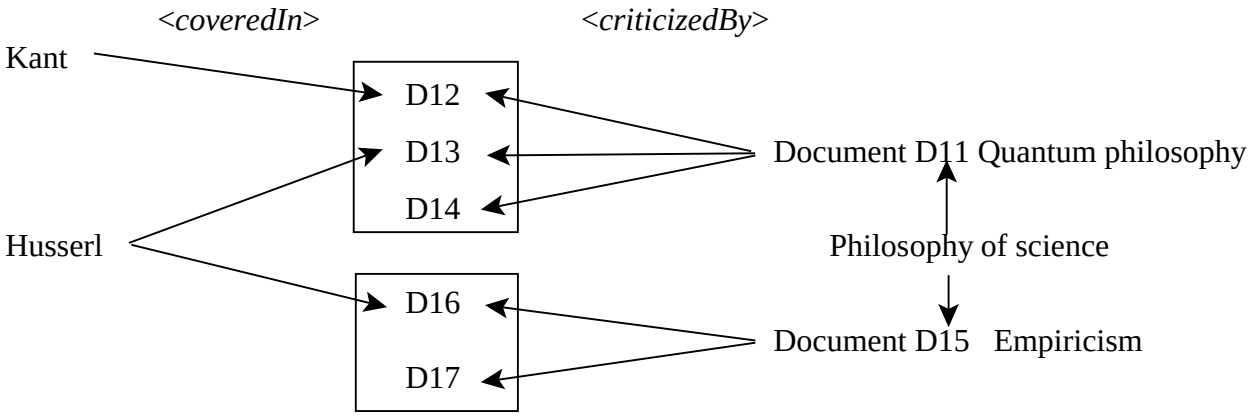


Figure 27. **Combination search (Boolean AND) with neighborhoods as targets. Example 3.**



Example 3 (Figure 27, complex)) illustrates the power of the general approach. The user is interested in a comparison of Kantian and Husserlian perspectives with respect to philosophy of science. A direct search did not find anything. But a data mining approach exploiting the information inherent in the links might find some documents that taken together would shed light on the subject. If we find a book in this subject area (Document D11) and this book was critiqued from a Kantian perspective in Document D12 and from a Husserlian perspective in Document D13, reading both D12 and D13 might shed light on the Kant-Husserl comparison. A search in several steps does the trick. First search on *philosophy of science*, resulting in a set of documents including D11 and D15. To each of these documents corresponds a neighborhood consisting of all other documents reachable through <*criticizedBy*>, such as D11 <*criticizedBy*> **D12**. These neighborhoods are the targets of a second search, a combination search for Kant AND Husserl following <*coveredIn*>. The search finds a neighborhood of critique documents – a neighborhood bound together by the critique of the same document – whenever it contains at least one document mentioning Kant and at least one document mentioning Husserl.

Specifying neighborhoods as search targets requires definition of the kind of neighborhood desired, such as offspring neighborhoods starting at whole document nodes or citation neighborhoods starting at any document following citation links one step (or two steps, or n steps). Defining target neighborhood types specification of a search pattern: Instead of specifying an individual starting object, specify a starting object type and the connection type(s) to be used to reach other elements of the neighborhood. This search pattern is repeated from each value of the starting object type to generate all possible target neighborhoods.

Present hypertext systems do not allow the specification of neighborhoods as search targets. Thus it is not possible to conduct a whole-document level Boolean search even if Boolean searching is implemented. Many hypertext systems allow Boolean searching for an "index search" but not for a search using the typical hypertext links. In bibliographic IR systems the search level supported depends on the type of descriptor used. Subject descriptors assigned through explicit indexing are assigned to documents as a whole. With text words, one can specify as the search target a whole document or a paragraph or a sentence. Another application of this concept is the specification of units for IDF computations or LSI.

4.6 Offspring and ancestor neighborhoods in searching

An offspring neighborhood (Section 3.3.1) is the neighborhood of all the objects reached from some top object following some hierarchical relationship down, such as a book with its chapters, sections, and paragraphs. Offspring neighborhoods provide an important structuring mechanism (see Section 3.3.1) and are also important in searching (see Section 4.6.1).

An **ancestor neighborhood** is the converse of an offspring neighborhood: the neighborhood of all objects that can be reached from some bottom object following some hierarchical relationship up. Examples: a paragraph in a book or journal article, the section containing the paragraph, the chapter, and finally the book or article itself; the concept neighborhood assembled starting from *thinking* following Broader Term to *cognitive processes* and finally *cognition and memory*, thus {thinking, cognitive processes, cognition and memory}. Ancestor neighborhoods are of importance primarily in searching, but they are also useful for display: When a given object, such as a document section, is found, displaying the ancestor neighborhood provides context.

4.6.1 Offspring neighborhoods and searching. Review

An offspring neighborhood can be used as a starting point in a search (Section 4.4). An offspring neighborhood of concepts implements inclusive (hierarchically expanded) searching, searching for a descriptor and all its narrower descriptors (*explode* or *cascade*, in DIALOG indicated by ! following the descriptor). An offspring neighborhood of assertions supports a thorough search for evidence disproving the assertion: In a search for counterexamples that disprove a general mathematical theorem, look for counterexamples for any of the special cases as well.

An offspring neighborhood can also be used to specify the search targets, thus defining a scope for a Boolean search condition (Section 4.3.2 and Section 4.5.2). A searcher can specify that two subject descriptors must co-occur in the same section (a very strict condition) or in an entire article (the neighborhood of all sections included in the article, a much looser condition).

These examples illustrate the power of the general concept of an offspring neighborhood; one formalism handles three seemingly different situations.

4.6.2 Ancestor neighborhoods and searching. Hierarchical inheritance

An ancestor neighborhood can be used as the starting point in a search to find all objects connected to a given object and its superordinates, such as all documents on thinking and the concepts in the hierarchical chain above it:

{thinking, cognitive processes, cognition and memory}

More important is the use of ancestor neighborhoods as search targets; they provide an approach to hierarchical inheritance. Consider a journal article dealing with AIDS. A section in that article might say "Five patients were treated with ... . Three patients responded to treatment in 7 days ... ". A search for document sections on the treatment of AIDS would not find this section since AIDS can only be seen from the context of the document as a whole. So the system must consider the link to AIDS from the node for the document as a whole; the concept AIDS should inherit down to the individual document sections. To give another example: A user is interested

in an **explanation of drug treatment of AIDS for the educated lay person**. Retrieval must consider not only subject matter but also the intended audience. The intended audience can often be seen from the journal, such as *Scientific American*, in which an article appears. All articles in *Scientific American* should inherit the attribute <*hasTargetAudience*> Educated lay person.

Figure 28 combines both examples to illustrate hierarchical inheritance two steps down. By defining ancestor neighborhoods as search targets, the Boolean search considers not only the concepts linked to the document section but also the descriptors linked to the journal article as a whole and to the journal as a whole, thus implementing hierarchical inheritance.

Figure 28a. **An ancestor neighborhood seen from the document perspective**

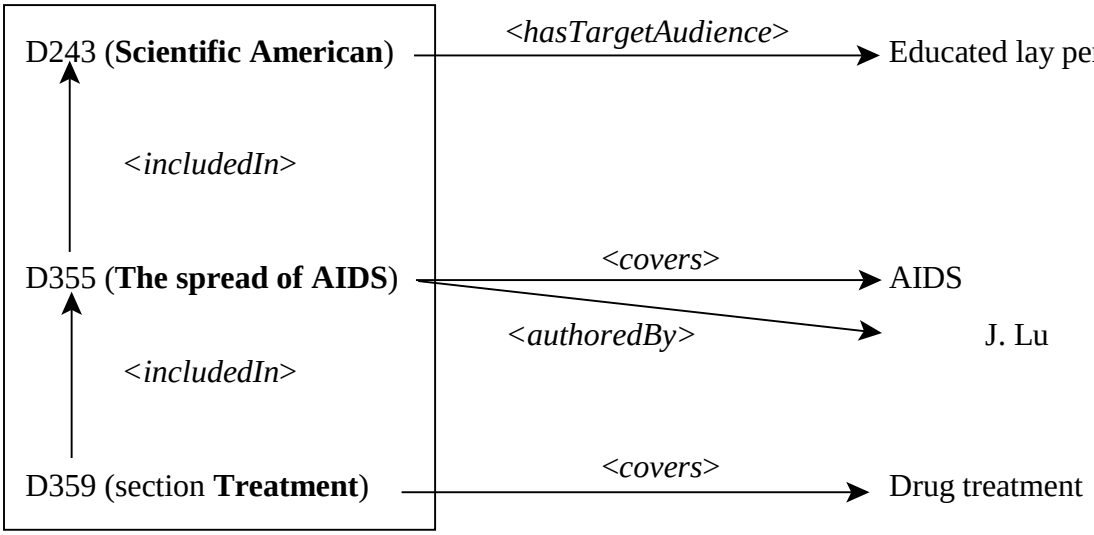
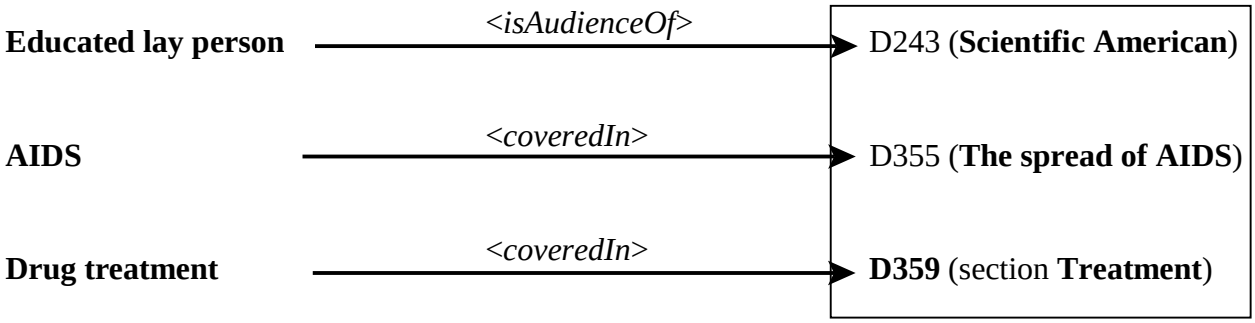


Figure 28b **Hierarchical inheritance through ancestor neighborhoods as search targets.** (Using the ancestor neighborhood from Fig. 28a, now seen from a search perspective.)



The sample ancestor neighborhood fulfills all three search criteria.

4.6.3 **Indexing with hierarchical inheritance**

Because of its close linkage with searching, indexing with hierarchical inheritance is discussed here, rather than in Section 5. When an indexer works on an object, such as a document, that has multiple objects under it, he must decide at which hierarchical level a relationship should be made. He should use a principle known from semantic networks and frame systems: If a relationship applies to all or almost all subordinate objects, establish the relationship at the superordinate object as an inheriting relationship. If the relationship applies only to a specific object, establish the relationship only for the specific object. This principle ensures parsimonious indexing with no ill effect for searching, provided hierarchical inheritance is applied, for example by specifying ancestor neighborhoods as targets as described in Section 4.6.2.

As a first example consider Figure 28a. Here we have a hierarchy of documents: a journal, an article, a section of the article. A statement that holds for all objects (e.g., all sections) under a superordinate object (e.g., an article) should be made at the superordinate level. In the example, all articles included in the journal *Scientific American* are geared to the educated lay person, so the statement <*hasTargetAudience*> Educated lay person should be made at the journal level. All sections of the article D355 deal with AIDS, so the statement <*covers*> AIDS should be made at the article level. On the other hand, treatment is the topic of a specific section, so the statement *covers* Treatment should be made at the section level. Stated differently: The following statements are true for the section (D359):

<*covers*> Drug treatment,

<*covers*> AIDS,

<*hasTargetAudience*> Educated lay person.

However, only the first is unique to the section, the other two are already included for superordinate objects. Thus there is no need to repeat them; only one statement needs to be added.

Figure 29 gives more examples. The second example deals with a hierarchy of descriptors; it is a restatement of a well-known indexing rule: Rather than assigning several specific descriptors that are all children of the same broader descriptor, assign the one parent descriptor that includes them all.

The third example also deals with a hierarchy of descriptors, this time in a thesaurus-building context. Rather than giving the same <*RT*> (<*hasRelatedTerm*>) relationship for several specific descriptors that are all children of the same broader descriptor, give the <*RT*> relationship for the one parent descriptor that includes them all.

Figure 29. **Indexing with hierarchical inheritance. Examples 2 and 3**

Example 2: Considering the descriptor hierarchy in descriptor assignment

Assume the following statements are all true:

Document D8 <*covers*> comprehension / comprehension <*covers*> Document D8

Document D8 <*covers*> cognitive mapping

Document D8 <*covers*> . . . (serial ordering, associative processes, mental concentration, ideation, thinking)

Using the common parent of all these concepts, one statement will do to sum it all up:

Document D8 <*covers*> cognitive processes

Accordingly, if cognitive processes <*coveredIn*> Document D8 is true, <*coveredIn*> inherits down to comprehension, cognitive mapping etc. Thus,

thinking <*coveredIn*> Document D8 is true through inheritance.

A complete search for thinking should start from the ancestor neighborhood

{thinking, cognitive processes, cognition and memory}

Example 3: Considering the descriptor hierarchy in establishing thesaurus relationships

Each of the descriptors under cognitive processes is related to intelligence. But instead of eight relationships comprehension <*hasRelatedTerm*> intelligence, cognitive mapping <*hasRelatedTerm*> intelligence, etc., we should establish just one inheriting relationship

cognitive processes <*hasRelatedTerm*> intelligence

5 Indexing

Throughout this paper it was assumed that there is a database with a plethora of objects and links. Of course, such a database must first be created. Somebody or some program must enter the objects and create explicit links between them; we call this process **indexing** in the most general sense. Objects and relationships can be created by authors, by specially appointed editors/indexers, by computer programs, and by users. The following examples illustrate the range.

The author of a whole document creates <*includes*> and corresponding <*continuedBy*> links from the top node for the whole document to each section, and from each section to paragraphs and figures. In a printed document (or the corresponding word processor file) these links are presented through the physical arrangement of sections in the whole document and of paragraphs in each section. To format the document for a hypermedia base, she must provide explicit links. (When a traditional document is to be included in a hypermedia base, the relationships implied by the arrangement must be converted to explicit links by the system or an editor.) The author may also create a <*hasAbstract*> link to another document. By her choice of words, the author also creates a link between each word in the text and the text section in which the word occurs; in an index for free-text searching these links are made explicit by a computer program. Specially appointed editors and "indexers" (in the common, much more narrow usage) create further links; for example, links between subject descriptors and a document as a whole (the most common form of "subject indexing") as well as links between subject descriptors and individual document sections or even individual paragraphs. An indexer may introduce (create) subject descriptors and create links between them, in which case the "indexer" is perhaps better called "thesaurus builder". Many might not even call establishing links between subject descriptors "indexing", but it is establishing links and requires intellectual decisions. Having stated the fundamental sameness, we should add that there are also differences: establishing links between descriptors does involve a different link type, requires a different type of thinking, and has more far-reaching, system-wide consequences.

There are explicit relationships that are stored as such in the information base and implied (computed) relationships derived through inference, similarity computation, or by the determination of an optimal reading sequence. Once implied relationships are derived, they can be added to the database as explicit relationships.

Part 2. Extensions and refinements

6 Introduction to Part 2

Part 2 begins with a more thorough discussion of the unified view of different types of systems, contrasting it with the view that considers these systems as fundamentally different and arguing the unified view and its advantages in detail. Section 8 then elaborates on the information structure, discussing additional features – including the representation of metadata – that increase representational power and search possibilities but also add complexity. Expanding on Section 4, which was restricted to the simplest form of search, search as navigation, Section 9 gives an overview of all search types. Section 10 elaborates briefly on indexing. Finally, Section 11 addresses design issues that arise in the implementation of the proposed system.

7 A unified view of systems (expanded from Section 2)

7.1 The multidimensional design space for information structure management systems

The approach discussed in this paper draws its power from its unified view of database systems, expert systems, information storage and retrieval systems, and hypermedia systems. This unified view provides the user with one set of search tools for all types of searches. This section concentrates specifically on a unified view of information retrieval and hypermedia systems and presents a refined analysis of the dimensions along which searches and entire systems can be analyzed (Figure 30, a refinement of Figure 7); these dimensions span a **design space**.

A1 Type of starting object(s). A typical hypermedia search starts from a paragraph, a picture, or an audiovisual object and follows links to other such objects. A typical bibliographic search starts from a subject or author and follows links from these to whole documents.

A2 Role of the starting object in the entire search and A3 Method of finding good starting objects. In the typical hypermedia search, the user looks at objects that actually provide some of the information needed and uses these same objects as starting points for the next search step. In a bibliographic IR system, the starting object – a subject descriptor or author name – is not of interest in and of itself but only as a means of finding target objects. The user must deliberately search her own memory or a thesaurus to find useful subject descriptors to start the search. In the typical hypermedia search, the user never worries about finding starting objects because the items of information she reads or looks at will also lead to further interesting items. In the typical bibliographic IR system, finding the right starting objects is a major concern.

B Method of specifying the starting object(s) and link types: either (1) clicking on elements displayed on the screen (often a default value) or (2) keying in their identifiers. From a practical standpoint, the ability to select choices by highlighting values displayed on the screen is essential for attaining the functionality of hypermedia systems, but it is neither theoretically necessary for hypermedia systems nor limited to them. Selection by clicking is available in modern

bibliographic systems: With a document record displayed, the user can click on an author name or subject descriptor in the record and thereby initiate a new search. This is a matter not of the basic system nature but of the user-system interface.

C1 Type of target objects found. The typical hypermedia search leads to the **media objects themselves,** a paragraph, a picture, or the full text of a document. The typical bibliographic search leads to **references to whole documents**; the user must find the actual text. But some systems first retrieve references and then, in a second step, provide access to full text. (There may or may not be links, such as citation links, between documents.)

On the other hand, a hypermedia system usually does contain nodes that are bibliographic references, such as an abstract of a whole document, the whole document itself being stored in the hypermedia base as a group of smaller media objects.

Many databases or information systems deal with objects that cannot be stored as such in electronic form, such as food products, organizations, or persons. In that case, an electronic information system can give only references. (From a more general point of view, a storeroom in which food products or technical parts or whatever are kept in an organized fashion is a retrieval system, albeit not an electronic one.)

C2 Granularity: Size of objects looked for or found. Granularity varies from very specific access to individual paragraphs, images, or sound packages (such as a passage within a piece of music) to access only to whole documents (books, book chapters, journal articles, etc.). This marks the difference between the typical hypermedia systems (and full-text retrieval systems, such as systems that search the full text of laws but may or may not include hyperlinks) and typical bibliographic retrieval systems, such as library catalogs or abstracting/indexing services.

The following four dimensions are expressions of the degree to which a search is *interactive*.

D1 The number of search steps in an entire search (information seeking episode). The prototypical hypermedia search has many search steps, the typical bibliographic search just one.

D2 Number of objects looked for or found in one search step. In the typical hypermedia search, the user is looking for just the next object to examine. The search following links from the object being examined generally leads to one or a few other objects. In the typical bibliographic search, the user is looking for a list of references, perhaps 10 or 20, to be used as the final answer set; a search may retrieve many more.

D3 Number of objects examined upon retrieval before moving on to the next search step. In a hypermedia search the user often examines only the first of several retrieved objects and immediately uses that object as the starting point in a next search step. The user of a bibliographic search typically selects and examines a number of relevant documents from the retrieved set and initiates additional search steps only if the information received is not sufficient or points out another information need.

Figure 30. **Dimensions for analyzing searches and systems** (expanded from Figure 7)

Dimension	**Typical hypermedia search**	**Typical bibliographic search without interaction**
A1 Type of starting object(s)	A paragraph, a picture, an audiovisual object	A search key: A subject descriptor, person, organization, etc.
A2 Role of starting object in the entire search	Mostly objects of value in their own right	Mostly objects used only for searching
A3 Method of finding good starting objects	Natural encounter during a search	Deliberate selection
B Method of specifying starting object(s) and link types	Object currently examined as default starting object Selecting link type from display on the screen	Entering elements from the keyboard
C1 Type of target objects found	Full text, picture, etc.	References to documents
C2 Granularity: Size of objects looked for or found	Individual paragraphs, images, or sound objects	Whole documents
D1 The number of search steps in an entire search	Many	Few
D2 Number of objects looked for or found in one search step	Few (1 - 5)	Many (10 -20 and up)
D3 Number of objects examined upon retrieval before moving on to the next search step in an entire search	Few	Many
D4 Role of the results of a single search step in the entire search	Piece of a mosaic being built Stepping stone for further searching	Final answer set of documents to be read
E Completeness and complexity of search specification for each search step	Partial, often implicit Simple	Complete, often carefully worked out Complex
F Ability of the user to augment the data base	Often built in as an essential part of the system	Usually non-existent

D4 Role of the results of a single search step in the entire search. What does the user do with the search results? She may consider the results of one search step merely as a piece of a mosaic being built through many steps – the berry-picking approach to assembling information, typical of (but by no means exclusive to) hypermedia searches. (The term berry-picking was actually coined by Bates 1989 in a paper suggesting a highly interactive and incremental approach for **bibliographic** retrieval.) Or the user might use the object(s) found as stepping stones for further searching, without or in addition to using them as sources of information in and of themselves, also fairly typical of hypermedia searches. Or the user might look at the results of a search step as the final answer set giving all the needed information, the system having done all the work assembling the answer without assistance from the user interacting with the system.

E Completeness and complexity of search specification required in each search step. The definition of a search step developed so far – starting from a single starting object and selecting all target objects reachable by a given type of link – is oversimplified. While this simplicity will do for some searches, others use a much more complex search specification, such as a Boolean query formulation or a query formulation using a relationship with multiple arguments (see Section 9). Devising such a query formulation requires effort. The typical bibliographic IR system works best with a complete and carefully worked out query formulation. In the typical hypermedia search the user is not even aware that she uses search specifications as she selects a link type for a search starting from the object being examined. A search may consist of one very complex step or of many simple steps.

F Ability of the user to augment the data base. Some hypermedia systems allow the user to add new objects and new links, either public or private; searching and the user's own work of reading, note-taking, commenting, and writing are not separate activities but all integrated in the interaction with a hypermedia system. Allowing user feedback to update and improve the database would also be a great benefit in bibliographic retrieval but is much less common there.

The dimensions discussed are to some extent independent from each other. For example, the *Information Navigator* by IME is a bibliographic retrieval system with a "hypertext feel" to it. (It is advertised as a hypertext application.) Like any bibliographic IR system, the *Information Navigator* displays bibliographic records consisting of a number of fields on the screen. (The underlying data store is arranged according to the entity-relationship approach, but that is not important for the discussion here.) Unlike other bibliographic IR systems, the *Information Navigator* lets the user select any object (person, organization, subject, series title) displayed on the screen as the starting point for a new search. Thus, if the user found a document in a subject search and wants to find more documents by the same author, she merely needs to highlight the author's name on the screen and press the search key. If one of the documents thus retrieved has an interesting subject descriptor, the user merely highlights it and presses the key for a thesaurus search to find broader, narrower, and related descriptors. Merely highlighting one of these and pressing the search key starts another subject search.

7.2 The functions of objects and relationships

This section elaborates on the various functions objects and relationships serve in structuring and searching information, strengthening the unified view.

7.2.1 The five functions of objects

Figure 31. **The five functions of objects (entities, nodes)**

An object can serve one or more of these functions

1 Participate in relationships that model actual data

2 Represent data for assimilation by users (documents)

3 Serve as an access point that leads to other objects

Example: Subject descriptor
But: Entry for subject descriptor also useful in itself, as in a dictionary.

4 Provide a focus for the organization of the database

5 Serve as focal points for relationships that pertain to all elements of a neighborhood (inheriting relationships)

First, objects participate in relationships that model actual data. One can make a statement about chocolate chip cookies only if *chocolate chip cookie* is an object in the information base.

Second, objects represent data for assimilation by users. A paragraph, a figure, a sound document are meant to transmit information, to enter the user's cognitive or affective sphere.

Third, an object may serve as a starting point or a query element to access other objects; for example, a concept or a person can serve as access point to documents, software objects, or organizations. Some objects, such as concepts to be used as subject descriptors, are introduced primarily for the access function. However, information about concepts is also useful in itself: Definitions may be of interest, and thesaural relationships (Broader Term, Narrower Term, Related Term) contribute to the definition and otherwise convey information, thus helping a user whose final information need is clarification of the meaning of a term. This illustrates a general point: An object can, and often does, serve several functions simultaneously.

Fourth, objects provide a focus for the organization of the information base; this function is very important for hypermedia bases. The head of an offspring neighborhood, such as the top node for a book leading to all the chapters, or a path object serve this function.

Fifth, objects serve as focal points for relationships that pertain to all elements of a neighborhood (inheriting relationships). For example, the relationship Scientific American <*hasTargetAudience*> Educated lay person is introduced with the idea that it holds for all the articles <*includedIn*> Scientific American (hierarchical inheritance).

7.2.2 **The two functions of relationships (links)**

Figure 32. **The two functions of relationships (links)**

Modeling actual data

Examples:

FoodProduct 1 <*hasIngredient*> [FoodProduct 2, ...]

Data set <*supports*> Assertion

Pointing to other objects

Most links

Examples for links established primarily as pointers:

Document <*includes*> Document

Document <*continues*> Document

Concept <*coveredIn*> Document

First, relationships serve to model data by connecting objects to construct statements. The <*hasIngredient*> relationship for food products is an example. Another example is DataSet <*supports*> Assertion, a factual statement made possible by including the relationship type <*supports*> in the conceptual data schema.

The second function of relationship or links is to point to other objects. Most relationships established to model actual data can also be used as pointers: On the one hand, I may want to know the ingredients of chocolate chip cookies; I use the relationship <*hasIngredient*> for its substantive information value. On the other hand, I may want to use this relationship to find all foods containing chocolate chips, in which case I use it as a pointer for retrieval purposes. Links between documents are established primarily for their pointer value. Document D355 <*includes*> Document D359 does make a statement about these two objects, but the main purpose of the link is to point the reader of Document D355 to Document D359.

The observation that objects (entities) and relationships (links) serve multiple functions, some more familiar in the database world and some more familiar in the hypermedia world, underscores the advantage of the unified view presented in this paper. It shows that existing systems set up primarily to store factual data can be navigated (provided proper software support) and existing systems set up primarily for navigation can be used to look up facts.

8 Elements of a unified information structure (augmenting Section 3)

The subsections of this section present features added to the information structure. They can be read independently from each other.

8.1 Defining schema neighborhoods – Modeling the semantic structure of a document

Schema neighborhoods are useful for the creation and display of structured documents, such as legal cases (Halasz 88, p. 843). As the example in Figure 33 shows, a schema neighborhood consists of a head object of a given type (in the example the type legal case) together with nodes related in one of several ways. A system can support the creation of structured documents by asking the user for the type of document to be created and then displaying the appropriate schema template. It can support the display of structured documents by showing the schema outline when the head node is displayed. The relationship to frames is evident.

Schema neighborhoods share many characteristics of offspring neighborhoods (Section 3.3.1)

Figure 33. **Schema neighborhood example: Legal cases**

Head node for the legal case as a whole

The schema neighborhood for a legal case includes nodes related in one of the following ways:

Legal case *<dealsWithFacts>* Object

Legal case *<dealsWithIssues>* Object

Legal case *<reachedDecision>* Object

Legal case *<givesRationale>* Object

Each of these four objects is the head of an offspring neighborhood containing nodes for the individual facts, the individual issues, the elements of the decision, and the elements of the rationale, respectively. Issues can be represented as descriptors from a classification of legal issues or by a textual description or both.

8.2 More on connections

Connections were introduced briefly in Section 3.2 as a means to extend a search along a chain of links to lead from the starting point to search targets. Connections add considerable power to searching; named connections are equivalent to simple Prolog rules (see Section 8.2.1).

As an example, consider the connection

[Greek vase <*hasInstance*> PhysicalObject-1, PhysicalObject-1 <*depictedIn*> Slide-567]

It leads from the descriptor Greek vase to a slide depicting one. The **connection type** is

[Concept <*hasInstance*> PhysicalObject, PhysicalObject <*depictedIn*> Slide].

A connection type is defined as a chain of links, each link belonging to a specified link type. We include "chains" consisting of a single link, called **direct connection**; a connection consisting of two or more links is called an **indirect connection**. For examples see Figure 34.

An indirect connection can be identified by a name, and that name can then be used just like a simple link, except that the system – transparent to the user – follows several steps.

Often an indirect connection has a meaning equivalent to (serves the same or similar purpose as) a direct connection. For example,

[Person P <*affiliatedWith*> Organization O,
Organization O <*hasPhone*> PhoneNumber N]

is for many searches equivalent to (can be substituted for)

Person P <*hasPhone*> PhoneNumber N

The system should tell the user who employs a direct connection about equivalent indirect connections that offer added possibilities for reaching relevant objects. To do so, the system must know such equivalencies and other relationships between connection types.

Named connections provide a quick way for specifying a search step. Equivalencies between connections let the system expand the user's search specification. This is an application of expert knowledge stored in the system to assist the user.

It is often convenient to use a connection in the definition of a still more complex connection; thus on the most general level connection is defined recursively as a link or a chain of connections.

Figure 34. **Connections and relationships between them**

Sample connection type

[Concept <*hasInstance*> PhysicalObject, PhysicalObject <*depictedIn*> Slide]

Direct and indirect connections

A **direct connection** connects two objects through a single link

Examples

Connection 1: [Document D1 <*commentedBy*> Document D8]
Connection 2: [Document D1 <*criticizedBy*> Document D9]

Connection 3: [Person P <*hasPhone*> PhoneNumber N]

An **indirect connection** connects two objects through two or more links via intermediate objects

Examples

Connection 4: [Document D1 <*proposes*> Theory T,
Theory T <*commentedBy*> Document D10]

Connection 5: [Document D1 <*proposes*> Theory T,
Theory T <*criticizedBy*> Document D20]

Connection 6: [Person P <*affiliatedWith*> Organization O,
Organization O <*hasPhone*> Number N]

Relationships between connections (knowledge used for intelligent search support)
(For a fuller treatment of metadata see Section 8.6.)

Connection 2 <***isa***> Connection 1

Connection 5 <***isa***> Connection 4

Connection 4 <***equivalentTo***> Connection 1

Connection 5 <***equivalentTo***> Connection 2

Connection 6 <***equivalentTo***> Connection 3

8.2.1 **Connections as rules**

Consider the connection

[Person P *<affiliatedWith>* Organization O,
Organization O *<hasPhone>* PhoneNumber N]

We could assign to this connection the name

Person *<hasBusinessPhone>* PhoneNumber

with the effect that the user could use *has-business-phone* like a single relationship, and the system would use the connection to reach the desired objects.

This is equivalent to the Prolog rule

Person *hasBusinessPhone* Number :-
Person *affiliatedWith* Organization,
Organization *hasPhone* Number.

The connection name is the head of the rule, the individual links are the conditions.

8.2.2 **Complex connections**

The definition of some quite natural connection types requires more than just chaining different link types together. One may want to repeat a link type several times in the chain, as in example 1 in Figure 35. We could, of course, have written the *<hasRelatedTerm>* link twice, but the syntax in Figure 35 is shorter and more elegant. Example 2 illustrates the more important case where one wants the system to follow a link type an unknown number of times until an end point is reached (in technical terms: compute transitive closure.) For example, when looking for all objects that are hierarchically below (offspring) or above (ancestors) the system must follow an unknown number of *<includes>* or *<included-in>* links. In a language like Prolog one would use a recursive rule to handle this situation. The syntax proposed here, substituting the "wild card" character * for a fixed number, is intuitive even for the less sophisticated user.

An element in a complex connection definition could be single link type, such as *<includes>*, or it could in turn be a connection.

Instead of insisting on a single link type at a given place in the chain, one could allow for several link types to be followed in parallel. That is, link types and connection types can be assembled into neighborhoods of connection types called **bundles**. Any link or connection type in the bundle will do to lead from one object to another. Such bundles can again be named, and a good system would provide built-in bundles with names that the searcher could use without knowing the details of the definition. For example, a system might provide the bundle

<producedByExpanded> =
{<producedBy, <authoredBy>, <compiledBy>, <editedBy>, <illustratedBy>, <translatedBy>}

Figure 35. **Specifying connections with repeating links**

Example 1: Fixed number of repetitions

[Concept-1 <*hasRelatedTerm* Concept-2 (2)]

Starting from Concept-1, this connection leads to other concepts that are one or two steps away following the <*hasRelatedTerm*> relationship.

Example:

learning <*RT*> cognitive processes, cognitive processes <*RT*> **intelligence**

Example 2: Repetition until an end point is reached

[Concept-1 <*includes*> Concept-2 (*)]

Starting from Concept-1, this connection leads to other Concepts included in Concept-1, an arbitrary number of levels down. (<*includes*> is a very general relationship type; in a thesaurus one would use the designation NT for Narrower Term.)

Example:

cognition and memory <*includes*> **cognition**
cognition <*includes*> **cognitive processes**
cognitive processes <*includes*> **comprehension**
. . .
cognitive processes <*includes*> **thinking**

[Object <*covers*> Concept-1, Concept-1 <*includes*> Concept-2 (*)]

This finds all Concepts connected with a known object, and then for each of these Concepts all Concepts included.

This would make many searches more convenient. This bundle is actually the offspring neighborhood of <*producedBy*> in a hierarchy of relationship types (see Section 8.7 on metadata).

Replacing a relationship type by a relationship bundle that includes it relaxes the search specification. One could relax the specification still further to the universal link type (any link type will do) and thus define a connection of a given length regardless of link types in the chain.

Do not confuse connections with paths to be discussed in the Section 8.3. A path specifies a sequence of objects (or nodes) for perusal by a user, it defines one of the myriad possible sequences and thus relieves the user of the burden of determining at every turn which way to go. A connection on the other hand, defines a "jump" from a starting object to a target object, where the jump can be a single jump or a "multi-jump" following several links in a row.

8.2.3 **Weighted links**

Not all links of a given type are of equal strengths. A document may <*cover*> some concepts in depth and others only in passing. Some of a person's interests may be passionate and others only minor. An animal may feed mainly on plants but occasionally on other animals. Assigning a linking strength or weight to the link allows for a more faithful representation of these situations. In multi-argument relationships. a weight could be assigned to each argument place or slot. The linking strengths of a neighborhood-to-neighborhood link can be defined as a function of the number of element-to-element links. For example, the strength of the directed citation link form Journal A to Journal B could be defined as the number of citations from articles in A to articles in B, or as the sum of the weights of these citations. The linking strength of a connection can be defined in terms of the linking strengths of its constituent links or connections.

With weighted links neighborhoods become fuzzy sets. Linking strength can be used in computed relevance scores in ranked retrieval as discussed in Section 9.3.1.

8.3 **Paths and scripts**

Often a user is better served by reading a logically arranged sequence of documents than by hopping from node to node. A *stored path* prepared by an editor allows just that. Such a path corresponds to a traditional article or book. The same problem arises in writing computer programs. When a program is divided into a number of pieces, some orderings are necessary for compilation and additional orderings are helpful for understanding by a human reader. A user may also want to preserve her own path through the hypermedia base.

A path is a neighborhood in which <*continuedBy*> relationships induce a linear ordering of the objects in the neighborhood. There is one object for the path as a whole. This object contains the path name and may be linked to the creator, target audience, subject descriptors, etc. It must also be linked to the first actual object of the path. This object leads in turn to the second object through a relationship [Object-1, Path] <*continuedBy*> Object-2. Alternatively, the objects on a path could all be linked directly to the path node by a relationship Path <*includesWithSequence*> [Object, Sequence-number]. This relationship is a special case of <*includes*>; it imposes a sequence on the included objects. This is a detail of implementation of little concern to the user.

An object on a path can in turn be a path. Thus a book can be represented as a path of chapters, each chapter as a path of sections, and each section as a path of paragraphs and figures.

A **script** is an object that contains instructions that "orchestrate the display of other [objects]" (Halasz 1988, p. 846). A script guides the user through an information base, possibly in a very prescriptive manner. For example, a script might organize a programmed instruction sequence that draws on a hypermedia base. As the script is run, information is presented, the user is asked questions, and the next item of information presented is chosen based on the answers. Another script might have instructions for putting together a document using documents, retrieving data and calling a program to represent these data graphically, retrieving other data and applying a natural language generator, etc.; such a script could be called a virtual document.

8.4 Statements about statements or statements as objects

As discussed in Section 3.2, an information base consists of statements created by relating one or more objects through a given relationship type. Statements can themselves be objects that can participate in relationships, as in the following examples.

Document D18 <*disputes*> Statement [Data Set A <*supports*> Assertion B]

Somebody claims that Data Set A supports Assertion B, but Document D18 disputes that claim. The relationship of D18 is neither to the data set A alone nor to the assertion B alone but to the statement linking them.

Document D19 <*disputes*> Statement [Document D12 <*criticizes*> Document D11]

Again, the relationship is not from D19 to just D12 or to just D11 but to the <*criticizes*> link between the two.

An important use of the ability to make statements about statements is the expression of truth values, since the truth of statements in an information base is seldom absolute. Since the truth or surety of a statement may depend on whom you ask, a three-way relationship is required:

Statement <*isBelievedBy* [object, strength]

where the object holding the belief could be a person or the system itself (indicating the strength of the consensus belief held by the system builders).

8.5 Virtual objects: Links as program calls

When a target object is not available, it can often be generated. For example, a user may start from a given data set and look for pictures that represent it graphically. Suppose no such picture is found but that the system includes a graphics program that could generate one. The system must provide for a link from the relationship type <*graphicallyRepresentedBy*> to the graphics program, use that link to find the program, call the program with the data set as argument, and display the graphical representation on the screen. All this is transparent to the user; as far as the

user is concerned, there is a graphical representation in the information base, and she found it by following the appropriate link. Since there is, in fact, no such image stored, the image can be considered as a **virtual object** and the <*graphicallyRepresentedBy*> link as an **implied link to a virtual object** (implied since it is not stored explicitly but inferred from the type "data set" of the starting object).

Here is another example of a virtual object: If the relationship <*otherLanguageVersion/French*> does not find a document, the system could invoke a translation program. Note that these programs are objects or neighborhoods in the information base; there might be a node for the overall program and hierarchically lower nodes for the program modules, other nodes for documentation, etc. (see Section 8.6).

In a virtual link, the call to a program is implicit and transparent to the user. An explicit call to a program or function, either by entering the name or selecting from a menu, could also be handled by the general syntax of invoking a link.

8.6 **Modeling a programming environment**

Programming environments are now designed **as hypermedia/database systems** with exciting new possibilities. A brief sketch of a hypothetical system is included here to illustrate how the unified systems approach can be used to model the capabilities of modern CASE (Computer Assisted Software/Systems Engineering) systems and perhaps even improve their design.

A programming environment requires additional object and relationship types that open new ways of supporting the programmer (Figure 36):

- When the programmer modifies a program, the system can display all calls to that program (particularly handy where the change involves the parameters needed in a program call).
- When the programmer modifies a variable definition or when she modifies a program part that changes a variable value, the system can display all programs using that variable.

The most helpful part might be the linkage to documentation.

- Whenever a program part is changed, the system shows the corresponding parts of the documentation (both system documentation and user manual) or at least alerts the documentation editor.
- The following example illustrates even tighter integration. Programs and user documentation often include the same value lists (for example, a list of relationship symbols allowed in a thesaurus). Rather than maintaining such a list in both places and risking inconsistency, one should be able to maintain just one list with appropriate format conversion for program or documentation use, respectively.

- The programmer can search the program (function) library for programs serving a given purpose and select from the retrieved list directly into her program.

These are just a few examples of the support an information structure management system could provide to the programmer.

Figure 36. **Some additional object and relationship types for a programming environment**

Additional object types

Variable

Instruction

Program segment, function (module, routine), program file are all covered by object type Program. A hierarchy is established by <*includes*>

ProgramType (with values source file, object file, and executable file)

ProgramPurpose (such as sorting a list or extracting a substring)

ValueList

MenuText

Additional relationship types

Program	<*calls*>	Program
Program	<*serves*	ProgramPurpose
Program	<*defines*>	Variable
Program	<*changes*>	Variable
Program	<*uses*>	Variable
Program	<*affects*>	Program (This relationship can often be inferred.)
Document	<*documents*>	Program
Program	<*includes*>	ValueList
Document (part of documentation)	<*includes*>	ValueList

8.7 Metadata: Data about the data structure

Knowledge about the data structure is important for searching; it has far-reaching consequences for many searches. This type of knowledge enables the system to behave much more intelligently. We will discuss two kinds of knowledge about structure. The first deals with hierarchical and other relationships between object types and between relationship types, the second with specification of the circumstances under which hierarchical inheritance is valid.

We use a very specific and restricted definition of the term *metadata*: metadata are data about the data structure, about relationships between object types and between relationship types. Often the term is used much more loosely to refer to data about data-containing objects used for the purpose of finding or evaluating such objects. In our view, these data are not distinct in principle from any other kind of data; in fact, it is the use of a piece of data that makes it metadata in the loose definition.

8.7.1 Relationships between object types and between relationship types. Object types and relationship types as objects

The importance of knowledge about relationships between object types and between relationship types for searching is illustrated through the following examples.

Example 1. When a searcher specifies *TextDocument* as the target object type, the system should also look for the more specific object types *JournalArticles*, *GovernmentReports*, etc.

Example 2. When a user, after reading a paragraph, wants to find paragraphs commenting on it and thus specifies the relationship type <*commentedBy*> to lead to other paragraphs, the system should also use the more specific relationship types <*supportedBy*> and <*criticizedBy*>. In order to do this, the system must know the hierarchy of object types and the hierarchy of relationship types. In both cases, the system should assemble the offspring neighborhood of the object type or relationship type, respectively.

How to represent relationships between object types and between relationship types? On a formal level, these relationships are no different from relationships between any other object, and the system should treat them that way both internally and in the user interface. The operation to see the object types that fall under *Document* should be no different from the operation to see all objects included in a given book or all concepts narrower than *cognitive processes*. Thus the simplest way of representing these relationships is to consider object types and relationship types as objects and use the general information structure machinery.

Figure 37 and 38 give further examples of these relationship types.

Figure 37. **Relationships between object types**

JournalArticle	<*isa*>	Document
GovernmentReport	<*isa*>	Document
TextObject	<*isa*>	Document
Map	<*isa*>	VisualObject
VisualObject	<*isa*>	AudiovisualObject
SoundObject	<*isa*>	AudiovisualObject
AudiovisualObject	<*isa*>	Document
City	<*isPartOf*>	Address

Figure 38. **Relationships between relationship types**

Document <*criticizedBy*> Document	***<isa>***	Document <*commentedBy*> Document
Document <*praisedBy*> Document	***<isa>***	Document <*commentedBy*> Document
Person <*isAuthorOf*> Document	<***inverseOf***>	Document <*hasAuthor*> Person

Universal relation or link as the top of the <*isa*> hierarchy

In indexing: relationship type not specified

In searching: any relationship type acceptable

8.7.2 **Inheritance specifications**

Section 4.6 demonstrated the importance of hierarchical inheritance for efficient information storage and retrieval. So the system must know when to apply hierarchical inheritance, and the specifications for this are by no means simple. Figure 39 gives examples of hierarchical

inheritance specifications expressed in the general entity-relationship syntax used for all the data in the model.

Figure 39. **Hierarchical inheritance specifications**

[Document <*hasTargetAudience*> GroupOfPersons]	<*inheritsDownAlong*> <*includes*>
[Document <*covers*> Concept]	does not always inherit down (no statement stored in the database)
[FoodProduct <*hasConstituent*> ChemicalSubstance]	<*inheritsDownAlong*> <*isIngredientOf*> (if Egg <*hasConstituent*> Fat and Egg <*isIngredientOf*> Custard, then Custard <*hasConstituent*> Fat)
[FoodProduct <*underwentProcess*> Process]	<*inheritsDownAlong*> <*includes*> if the meaning is any of the ingredients underwent the process but not if the meaning is the food product as a whole underwent the process
[FoodProduct-1 <*isa*> FoodProduct-2]	does not inherit down along <*isIngredientOf*> (Egg <*isa*> UnprocessedFood and Egg <*isIngredientOf*> Custard do not imply Custard <*isa*> UnprocessedFood)
Any relationship whatever	*inherits-down-along* inverted-<isa> (with exceptions overriding the inheritance explicitly indicated in the database)

9 Search (augmenting Section 4)

Section 4 discussed one kind of search, search based on following relationships (links), and used the navigation metaphor to describe searches. But, as mentioned in Section 4.1, there are other types of searches. This section presents an overview and briefly discusses the other search types. Sections 9.1-9.3 form a logical progression, while Sections 9.4-9.6 are independent.

Figure 40. **Types of search**

Search based on relationships between objects (Section 9.1)

- Search based on using relationships from one or more starting objects to identify target objects (Section 4, Section 9.1.1)
  - Two perspectives:
  - Search as navigation (based on links, that is, binary relationships)
  - Search as query-based retrieval (based on relationships of any order)
- Similarity search based on relationships: Find objects with a neighborhood similar to a specified "query neighborhood". The query neighborhood can be specified as the neighborhood of a starting object. (Section 9.1.2)

Search based on intrinsic properties of objects (Section 9.2)

Search based on a relevance score rather than exact match (Section 9.3)

The big divide is between searches based on relationships between objects (Section 9.1) and searches based on intrinsic properties of objects (Section 9.2). (Of course, there are searches combining criteria from each.) Searches based on relationships do not look inside the objects (texts, images, sound clips). In some ways this is limiting. As an example, consider a search where the user draws an arbitrary shape and asks for images that contain a similar shape; this search requires an examination of the (digitally stored) images in the collection. As another example, consider a search that looks for texts that contain a specified syntactic pattern or that contain two words in a given syntactic relationships (or just in proximity to each other). Again, that requires analysis that looks into the texts. More generally, these searches are based on internal properties of objects or, with neighborhoods as search targets, on the structural properties of a neighborhood derived from the configuration of relationships within it.

Searches based on relationships can in turn be divided into those that use a starting object and follow relationships to target objects, using either the navigation or the query metaphor (Section 9.1.1), and those that are based on the similarity of the neighborhood surrounding an object to a query neighborhood (Section 9.1.2).

Section 9.3 introduces more complex search algorithms that achieve a ranking of retrieved objects by expected relevance, possibly through the use of weights. Section 9.4 discusses a number of refinements of search process: meaningful arrangement, assembling a result neighborhood, limiting the search to a specified neighborhood. Section 9.5 discusses further search refinements and Section 9.6 introduces the perspective of search as inference.

The systems described in Croft 1989 and Thompson 1989 illustrate several of the individual approaches that we discuss here in a general framework.

9.1 Search based on relationships between objects

9.1.1 Navigation and query-based retrieval as two perspectives on searching

Searches differ in the method used to get from the known to the targets. In the search examples given in Section 4, the method is navigation based on links (binary relationships). The user specifies his query by selecting an object on the screen as the starting object and selecting a type of link to follow. The same result could be achieved by formulating an explicit query, such as

Find all documents that have a <*covers*> link to AIDS

or, expressed in terms of field values

Find all documents with DE(scriptor field) = AIDS.

Of course the query could combine several conditions using Boolean operators, such as

Find all documents that have a <*covers*> link to AIDS AND to Drug treatment
AND a <*hasTargetAudience*> link to Educated lay person

More complex cases, particularly searches using higher-order relationships, **require** an explicit query. The distinction between search as navigation and search as query-based retrieval (Halasz 1988, p. 841) is more a matter of perspective than of the basic nature of the search; they are two different metaphors for the same process. The search procedures corresponding to these metaphors are different, but this is a matter of degree rather than absolute difference (see Sections 2 and 7). The nature of the starting object (for example, document vs. subject descriptor) may also play a role in determining the perspective. As the discussion and the examples in Section 4 make very clear, the principle is always the same: The user starts from an object or neighborhood and, following a given link type, finds other objects or neighborhoods.

The typical hypermedia search uses the navigation metaphor. The starting object is the document (a paragraph of text, a picture, etc.) currently on the screen, and the user expects to find just one, or at most just a few, documents to look at next. The document found then becomes the starting point for the next search step, and so forth. A system might facilitate this process by showing on the screen a "map" of the links between objects, an outline being a special case.

The typical bibliographic search uses the query metaphor. The starting object is a subject descriptor or an author name, and a link type is often specified. The user won't be surprised to to find 30 or 50 or even 200 documents. A query can also be more complex, combining several starting objects and requiring that a target be reachable from all. Query formulations are also required for searching data represented through relationships with many arguments, for building elaborate inferences into a search, or for deriving new data through processing. Such query formulations can get quite complex. Consider, for example, queries in a database query language such as SQL.

9.1.2 Another view of query-based search: Search for objects based on the match of their neighborhood to a query neighborhood

This section presents a different view of query-based searching, introduced first through an example. In a navigation search, a search on the concept of thinking was accomplished by specifying thinking as a starting point and following <*coveredIn*> links. Instead, we could instruct the system to find all documents whose <*covers*> neighborhood contains thinking. Or we could require the presence of two concepts in such a neighborhood, say thinking and alcohol. In this search the system is viewed as examining the neighborhood of each target object to see whether it meets certain criteria. The specification of these criteria is a query. The criteria for specifying the query neighborhood can include all types of direct or indirect connections, making possible complex searches that could not be accomplished by navigation. The user can also pick a known relevant object and request target objects that have the same neighborhood with respect to one or more connection types; for example, objects that are indexed by the same descriptors as the known relevant object. The relevant object, through its relationships with other objects, defines the query formulation.

9.2 Search based on intrinsic characteristics of objects or neighborhoods

So far, we have discussed searching based on the relationships or links between objects or neighborhoods. But selection of objects or neighborhoods can also be based on their internal structure or other properties. For example, free-text searching uses operators such as ADJacent, which requires that two words occur next to each other in the text (or no more than n words apart), or an operator requiring two words occurring in the same sentence. Free-text search systems that deal only with whole documents also provide an operator requiring that two words occur in the same paragraph. A hypermedia system that deals with paragraphs as objects does not need such a paragraph operator, since the search can just specify paragraphs as target objects. With parsers and computing resources both improving, we will soon see operators that allow the specification of syntax-based dependencies of words in text.

One can think of analogous examples in searching for pictures and for pieces of music. Assume a system that stores digitized images and that includes a picture recognition program capable of identifying subjects depicted, such as buildings and persons. One could then search for all pictures that depict a person inside a building. Similarly, with digitally stored music one could

search for all pieces of music that follow the form abba or that include a certain theme or a sequence similar to a theme.

The search targets can be atomic objects. Determining the internal structure of atomic objects requires programs, such as parsers, that can read the objects and process them to recognize the structure. (Halasz 1988, p. 842, calls this content search.) The search targets can also be neighborhoods. Determining the internal structure of a neighborhood requires a structure-recognition program that processes the links within the neighborhood. (Halasz 1988, p.842, calls this structure search.) Note that this is quite different from retrieval based on external links from or to a neighborhood.

An internal property search can use exact match or a similarity criterion (see Section 9.3).

9.3 Search based on a relevance score rather than exact match

The searches discussed so far are exact match searches; a target object is either found or not found. But a user will find some of these objects more relevant than others, and will judge even some of the objects not found somewhat relevant. So a more realistic approach is to have the system compute a score of expected relevance.

In a navigation search there are several mechanisms that can be used in such a computation. The user can vary the emphasis on starting objects and on the connections that lead to targets by assigning weights; this is particularly useful in the case where there are several starting objects used as alternate starting points (Boolean OR) or where there are several connections to be used in parallel. The system then might compute a relevance score as

query weight of starting point x query weight of connection x indexed weight of connection

(indexed weights for connections were discussed in Section 8.2.3). The weight of the starting point can further be modified by considering its intrinsic property of being useful for retrieval; inverse document frequency (IDF) has ben used as an indicator of such intrinsic usefulness. The computation might also consider the number of links in a connection, particularly in a connection that is defined with an arbitrary number of links (as defined in Section 8.2.1).

If there are several starting points such that target objects must be reachable from all (Boolean AND), the weight of a target might be computed as the sum of the weights determined from each of the starting points (which would result in a score even if the target object is reachable only from two of three required starting points.)

A search based on matching neighborhoods can be extended to a search resulting in relevance scores very naturally. We relax the condition that the neighborhood of an object must match the query neighborhood exactly and only require that the neighborhoods be similar in some way. Some possibilities for defining similarity between neighborhoods are given in Figure 41.

Figure 41. **Similarity search**

Starting from an object or a neighborhood, find objects or neighborhoods with similar surroundings or define a "query neighborhood" and find objects or neighborhoods whose surroundings are similar to this query neighborhood.

Two objects or neighborhoods have similar surroundings if the patterns of links emanating from them are similar. This could be restricted to types of links, for example

> Two documents (or other objects) are similar if they have links to many of the same concepts. Even better, the similarity algorithm could give some credit for related concepts, where concept relationships can be given explicitly or inferred by similarity as described below in the example immediately following.
>
> Two concepts are similar if they are linked to many of the same objects (put differently, if the sets of objects linked to each show considerable overlap).
>
> Two documents are similar if they share many cited documents.
>
> Two assertions are similar if contradicted by many of the same statements.

The possibilities for formulating similarity criteria are limitless. The user specifies the object to serve as the standard of comparison and the type(s) of connection(s) to be used in assembling the surroundings of the object. For a simple example, consider the neighborhood formed around an animal species by following relationships such as

Taxon <*hasSuperordinate*>	Taxon (several levels up),
Taxon <*livesIn*>	TypeOfEnvironment,
Taxon <*feedsOn*>	Object,
Taxon <*hasSize*>	MeasureNumber.

If the user simply specifies "similar", the system could use all direct connections to determine similarity. An important special case is a search for all objects that occur in a neighborhood of given structure in a given role. For example, one could look for all adjectives used as modifiers for an animal term.

The objects similar to a given object form a **similarity neighborhood**. Such a neighborhood could be used in any of the ways described above. For example the user could specify a Boolean AND search targeting similarity neighborhoods; that is, the user would instruct the system to select similarity neighborhoods that can be reached from both starting objects. However, such a search would require very extensive computation.

The principles for computing relevance scores based on the similarity of neighborhoods can be applied equally to search based on intrinsic characteristics or to search based on a combination of relationship-based and intrinsic criteria.

9.4 Arrangement of research results for ease of processing

Search results should be arranged to support processing by the user. There are many possibilities for such arrangement.

9.4.1 Arranging search results by relevance score

A very popular arrangement is a list ranked by relevance score, highest score first, so that the user can go down the list until she has found enough objects for her purpose. If the relevance scoring algorithm score reasonably matches relevance as judged by some expert or by the user (a big if), then relevance ranking should result in the user spending her time profitably in the upper part of the ranked list.

9.4.2 Meaningful arrangement of search results

Order by decreasing relevance is not always the most helpful order for the user. An arrangement by subject clusters, or by provenance of the objects found (for example, grouping documents by research project from which they emanate), to give two examples, may be more helpful. Still another approach is arranging objects in an order that makes sense for perusal, thus supporting the user in traversing the information base in a logical sequence. While some users might prefer to find their own way, others might welcome at least suggestions for a sensible path. A user could still detour from or completely leave the suggested path at any time.

Creating a logical path is difficult. Using a stored path or script is, of course, one solution. But a stored path is usually not optimally adapted to the needs of the individual user. The ideal system would consider the user's present state of knowledge, the state of knowledge required to solve the problem at hand, and then construct a path that leads from here to there. In an education context, the system could produce a "textbook" tailored to the background and learning objectives of an individual student. For example, one could envision an information base in statistics that would select the text pieces necessary to achieve an understanding of the statistics topics the user needs to learn about, considering the user's background in deciding both which topics need to be included and which version of parallel texts to select, select examples from the user’s subject field, and present the result as a tailor-made statistics textbook. A discussion of how this could be done is well beyond the scope of this paper, so a few hints must suffice. The system needs to rely on <*hasPrerequisite*> links to either select a document that transmits the information needed without requiring knowledge the user does not have, or, if that is not possible, include in the constructed path a document that provides the prerequisite knowledge. It must also consider something more subtle: The reading of one document may train the user in a way of thinking that is helpful in understanding another document; this kind of relationship establishes a basis for sequencing. The system must consider the user's background when choosing examples. The system must consider the user's cognitive style. For example, does the user learn best when examples are given first and general principles stated afterwards or when the general principles come first and are then illustrated by examples?

9.5 Other refinements of the search process

9.5.1 Assembling result neighborhoods during search

As the search proceeds, the user may mark items she wants to examine further or use for a given purpose, creating one or more **result neighborhoods**. This requires no new functionality. The user can simply define new objects (which could be called markers) corresponding to the result neighborhoods to be created and establish a link between a found object displayed on the screen and the appropriate marker(s). The system should facilitate this process. For example, the system could create temporary markers 1 - 9 which would exist until the end of the session or until a completely new search is started. Before the system erases a temporary marker and the associated links, it would give the user a chance to give it a permanent name. A marker may, in effect, become a descriptor that would be useful to other users as well.

9.5.2 Limiting the search

A user may limit the search to a neighborhood. That is, search steps executed after the limit is put into effect return only objects belonging to that neighborhood. For example, in bibliographic systems it is quite common to limit by year of publication. Or a user, having finished his search of the entire system, may now want to limit the search to the neighborhood of objects selected for further examination. Or the user may want to limit the search to objects included in a given handbook or journal, in other words, to an offspring neighborhood. If the neighborhood to which the search is limited can be computed by search (as opposed to being the result of selection by a user or other agent), then limiting is simply a convenient way of specifying a search criterion for all subsequent

A special case of limiting is the nesting of hypermedia bases: Allow an entire hypermedia base B as a node in a superordinate hypermedia base A (preferably allowing links between objects in A and B). When the user of A selects node B, she is placed into B for search (navigation) in B only. The subordinate hypermedia base B can be seen as a neighborhood in the superordinate hypermedia base A. In this approach, the neighborhood to which the search is limited is predefined; in limiting in general the user can define the neighborhood.

9.6 Search as inference

In the broadest sense, any search is the inference "if an object meets the search criterion (or criteria), it should be found". One is more likely to speak of inference when a search uses multiple criteria or when it uses a chain of links or relationships, thus combining multiple pieces of data to arrive at a conclusion about retrieval. For examples see the indirect connections discussed in Section 8.2. To give another example: One might introduce the connection

C <*maySupport*> A defined as the indirect connection [C <*criticizes*> B, B <*criticizes*> A]

Looking at search as inference greatly enhances the power of hypermedia systems. Including more elaborate inference would contribute to the expert system functionality of an information structure management system.

10 Indexing (augmenting Section 5)

Indexing – entering the entities / objects and relationship links into an information base – is a very expensive process when done manually. There are several strategies to cope with this problem:

- Build an integrated information structure interface on top of one or more existing databases.
- Convert the data from existing databases into a unified information structure management system. This becomes more challenging if the existing database is only moderately structured, such as converting from a machine-readable version of a dictionary (possibly generated by scanning the printed version) into an entity-relationship representation.
- Automate part of the indexing process. For example, there are systems that can convert a conventional text into a hypertext, isolating sections and subsections and determining linkages between sections beyond the <*includes*> links inherent in the text's original sequence and/or generating a table of contents.

These strategies make the creation of unified systems more feasible.

11 **Design issues**

The implementation of a general information structure management system that would combine the functionality of a database management system, an expert system shell, a retrieval package (including text retrieval) and a hypermedia shell, is a tall order. Such a system must solve two major design problems: It must provide a powerful search engine that can implement the search options discussed, many of which require intensive processing, and it must provide an interface that facilitates using the powerful general search operators. The system must also provide a basic set of object types and relationship types with proper semantics for processing built into the system and allow the user to define additional object types and relationship types with their semantics. Figure 42 lists these design issues.

Figure 42. **Design issues**

Provide a powerful search engine that can implement the search options discussed, many of which require intensive processing. This is difficult enough in a single system organized along information structure principles and even more complex in a distributed system accessing heterogeneous databases, where the search engine must be capable of issuing queries to many systems.

Provide gateways to existing information systems (databases, IR systems, hypermedia systems, expert systems) to obtain data needed in a search. Provide the capability for combining data from various sources.

Provide a flexible user interface

- General search operator syntax that works for all types of objects and relationships.
- Simplified search operator syntax for frequent special cases with preset values and default values.
- Menu-driven search specification with preset values and default values.

Certain types of neighborhoods (offspring, ancestor) available through proper notation without explicit search specification

Neighborhood schemas definable by users

Scripts

A basic set of object types and relationship types with proper semantics for processing built into the system

User-definable object types and relationship types with user-definable semantics

The search engine must use efficient algorithms for implementing the general search operators. Implementing such a search engine can be facilitated by using a powerful language specifically designed for processing data structured into frame and relationships, such as the Smalltalk Frame Kit (SFK) (Rostek and Fischer). The gateway function needs a capability for defining additional systems to be accessed.

As mentioned above, the system should come with a predefined set of object types and relationship types with defined semantics. But specific applications need their own definitions with additional object types and relationship types, with named connection types, named neighborhood types that are frequently used, and with neighborhood schemas. An information structure management system must allow for such definitions.

We conclude with a few remarks on the user interface. For navigation-type searches, the user need only specify the starting object (usually an object on the screen) and select the link type from a menu associated with the starting object. The search targets are often implied by the link type. Where appropriate, the system should remind the user that he could specify an offspring neighborhood as a starting point to broaden retrieval, and it should guide the user in the proper choice of targets (atomic objects or neighborhoods). The system should also suggest related link types or connection types to follow. The user should be able to refer to frequently used neighborhood types – offspring neighborhoods, ancestor neighborhoods, schema neighborhoods – by a simple symbol without formulating a query. If an offspring neighborhood could be formed along any of several hierarchical link types, the system should ask.

Once the system has constructed a query based on the user's answers, it should ask the user whether she wants to save it under a name for later use and possibly editing. The system also needs a syntax for query formulation by more experienced users. This language should incorporate all the capabilities of SQL and of a language like Prolog but not be limited to those.

Once an information structure management system with these capabilities is developed, it will put a powerful tool in the hands of a wide spectrum of users.

Notes

Some of the issues discussed in this paper have been mentioned or alluded to in Halasz' important 1988 paper, in which he discusses seven issues for the next generation of hypermedia systems and mentions the "unity of systems" (my term) theme (p. 847). Croft 1989 and Thompson 1989 also illustrate the integration of hypermedia and retrieval. Frisse 1992 reviews hypertext models with a view to integrating semantics and information retrieval features.

References

Bates, Marcia J. 1989
The design of browsing and berry picking techniques for the online search interface.
Online Review. 1989; 13(5): 407-424.

Carlson, David A. 1990.3; Ram, Sudah.
HyperIntelligence: The next frontier.
Comm. ACM. 1990.3; 33(3): 311-321.

Croft, Bruce W.; Turtle, Howard 1989
A retrieval model for incorporating hypertext links.
Hypertext '89 Proceedings. New York: ACM; 1989.11: 213-224.

Evans, Peter 1991
Personal communication.
Many of the ideas in Sections 2/7 were stimulated by conversations with Peter Evans, at the time a doctoral student at the University of Maryland.

Frisse, Mark E. 1988.7
Searching for information in a hypertext medical handbook.
Comm. ACM. 1988.7; 31(7): 880-886.

Frisse, Mark E. 1989.11; Cousins, Steve, B.
Information retrieval from hypertext: Update on the Dynamic Medical Handbook project.
Hypertext '89 Proceedings. New York: ACM; 1989.11: 199-212.

Frisse, Mark E.; Cousins, Steve B.
Models for Hypertext.
J. of the American Society for Information Science. 1992.3; 43(2): 183-191.

Halasz, Frank G. 1988.7
Reflections on notecards: Seven issues for the next generation of hypermedia systems.
Comm. ACM. 1988.7; 31(7): 836-852.
A seminal paper which provided an important takeoff point for the ideas developed here.

Fischer, Dietrich; Rostek, Lothar 1997
Smalltalk Frame Kit (SFK)
Darmstadt: GMD — IPSI; Draft, April 8, 1997
http://www.darmstadt.gmd.de/~rostek/sfkman3.1/index.htm

Swanson, D.R. & Smalheiser, D.H. 1997
An Interactive System for Finding Complementary Literatures: A Stimulus to Scientific Discovery.
Artificial Intelligence 1997; 91(2).

The Information Navigator
by IME (Information Management and Engineering)
990 Washington St., Dedham, MA 02026-6790
(617) 320-0303 FAX (617) 320-0793

Thompson, R. H.; Croft, W. B. 1989
Support for browsing an intelligent text retrieval system.
International J. for Man-Machine Studies. 1989; 30: 639-668.

Tompa, Frank Wm. 1989.1
A data model for flexible hypertext database systems.
ACM Transactions on Information Systems. 1989.1; 7(1): 85-100.