Forthcoming in:
IBM Internal Technical Liaison Conference: Expert Systems (October 19-21, 1992).

Extending Hypermedia with an Inference Language: An Alternative to Rule-based Expert Systems

Gerry Stahl, Ph.D. — Department of Computer Science, University of Colorado at Boulder, email: gerry@cs.colorado.edu

Raymond McCall, Ph.D. — College of Environmental Design, University of Colorado at Boulder, email: mccall_r@cubldr.colorado.edu

Gerri Peper, Ph.D. — IBM, Boulder, CO 80314

Abstract

Hypermedia systems can provide an alternative to expert systems in domains that call for navigation rather than inference. This paper reports on research to extend the hypermedia model to include an inferencing capability, so that the benefits of this approach can be gained in applications that require a mix of navigation and inference. An application in the domain of academic advising was developed to test this new approach and to compare it with the traditional hypermedia and expert system alternatives.

Comparing Rule-Based Expert Systems with Hypermedia

Expert systems are most useful in well-defined domains in which the rules can be made explicit. However a study by researchers at IBM (Peper, et al, 1990) identified a number of problems with traditional rule-based expert systems. The design of systems of rules is difficult and problem-laden. Even more of a concern is the issue of maintaining rule-bases. Maintenance is always a primary concern in the software lifecycle, as both the rules of the domain and the needs of the users evolve. Expert system shells, designed to obviate the need for specialists with computer programming expertise, have not eliminated these difficulties.

There are a number of reasons for the problems with rule-based systems. The first step, encoding and ordering the rules, is a major challenge. This is because the syntax of the rules is non-intuitive, and hence hard for users to understand and modify as well as awkward to encode. Furthermore, because of the nature of the inferencing process in the expert system engines, the ordering of the firing of rules is critical. The firing of rules can also have unwanted side-effects.

In particular, conflicts between which rules to fire can arise, creating what is perhaps the most significant problem for maintenance of expert systems: conflict resolution. In addition, even once the rules have been adequately debugged, special procedures often still need to be programmed in source code (e.g., Lisp). Finally, expert systems tend to be inflexible. They pursue a fixed line of inquiry entirely under the computer's control. Thus, it is not possible for the user to introduce new information unless explicitly prompted for it, or to explore the information in the system in an unrestricted manner.

The IBM study showed that hypermedia navigation could often provide an effective alternative to rule-based inference systems. Such an approach gives users greater control and allows them to explore the knowledge base. The study concluded that a hypermedia system could be just as effective and easier to create and maintain. Also, the hypermedia system can run faster and require less computer resources than rule-based expert systems.

The original idea led to HyperWin, an IBM product. Applications are, indeed, easy to construct, understand, and maintain with it. They can be used in an exploratory way with no training. Several applications have been developed in HyperWin, including an academic advising system for Auburn University.

Hypermedia represents an appealing alternative for situations in which all the knowledge can be laid out as a network of textual nodes and links for traversal by the user. However, there are many applications in which inference by the system would also be desirable or necessary. Hypermedia represents an appealing alternative for situations in which all the knowledge can be laid out for the user as a network of textual nodes and links. However, there are many applications in which inference by the system would also be needed.

Extending Hypermedia with Inferencing

We decided to add the power of inference to the elegance of hypermedia navigation. Some expert system applications can be defined as a network of nodes for navigating without inference, like .... Others are wholly reliant upon inference computations, such as .... But many applications fall between these two extremes and might best be served by combining the two approaches.

We began by building on an existing hypermedia system called Mikroplis (McCall, 1989), with an English-like query language that has been successfully used for several years. The Mikroplis query language is a comparatively simple language for navigating across links in a hypermedia document. It has a total of 12 syntactic options and a limited potential complexity, compared to over a hundred options in the inference language developed in this research. It allows the system to select links from a node and to check the content of nodes for the inclusion of a substring of characters.

To support a wide range of inferencing, the language had to be extensively expanded to include true/false conditionals, numerical calculations, comparison operations, and nesting of phrases.

(See the appendix for a listing of the abstract syntax of the new language, with the options from Mikroplis underlined.) A typical request in the new language -- taken from the test domain of academic advising -- might look like the following:

```
Display all courses of Sandra which have studio_types and
which also have less than 3 prerequisites, with their
prerequisites.
```

To evaluate this statement, the system would navigate from the student node, `Sandra`, across all its `courses` links; check which nodes arrived at had at least one `studio_types` link and also had less than three `prerequisites` links; and output a list of the course nodes that satisfied these conditions, along with a sublisting of their prerequisites. The output might look like this:

```
***COURSES
1. ENVD 2110  Architectural Studio
     *** PREREQUISITES:
     1. ENVD 1000  Environmental Design Studio
     2. ENVD 1014  Intro to Environmental Design
2. ENVD 2120  Planning Studio
     *** PREREQUISITES:
     1. ENVD 2110  Architectural Studio
```

The structure of statements in the inference language and their method of evaluation are based on the structure of hypermedia. The queries investigate the node and link structure, rather than the content of a database, and their evaluation proceeds by navigation across the links from initial nodes. In this sense, the research represents an effort within the hypermedia paradigm. The thrust of the effort is to exploit hypermedia mechanisms to achieve certain functionality of artificial intelligence and information retrieval technologies. Thus, the goal was to expand hypermedia to include:

* Some of the inferencing capability of Prolog, but without the comprehension difficulties of predicate calculus and explicit variables;
* Some of the querying ability of SQL, but without the inefficiency of relational joins;
* Some of the advantages of semantic databases, but allowing semantic relationships to be defined between instances as well as types; and
* Some of the utility of semantic networks, but without restriction to a pre-defined set of types.

A Navigation Language for Nodes

Our original approach relied heavily on the idea of *smart nodes*, in which the inferencing power is embedded in the nodes of the hypermedia. This was conceived primarily in terms of *virtual structures*, an extension of the fixed structures of textual or graphical nodes in traditional hypermedia systems, suggested by Frank Halasz (1988). We used the navigational (or structural) approach to query evaluation, as found in the Mikroplis language, and embedded the language in the hypermedia nodes. We did this to avoid simply gluing together two different paradigms (e.g., hypermedia and Prolog, or hypermedia and SQL, or HyperCard and HyperTalk) and to develop

the querying or inferencing capability out of the hypermedia paradigm itself ~~(as Frank Halasz (1988) and others in the hypermedia community had called for)~~.

The content of a smart node is not limited to the text or graphic originally entered into it. Instead the content is determined by the results of a query or conditional phrase associated with the node. The query traverses the hypermedia network, so its result depends upon the current state of the network: the existence of other nodes, their links and their current content. When smart nodes are displayed, the appearance of the hyperdocument itself changes dynamically.

Two forms of smart nodes were explored: *conditional nodes* and *virtual structures.* A conditional node contains a conditional phrase in the inference language and normal text or graphics. If the condition evaluates to true, the text is displayed. If the condition is false, nothing is displayed. For instance, in the academic advising application a node with the text, "`Are you interested in a studio course?`" might have the condition, `If there are courses which have studio_types`. Then the text would be displayed only if there actually were studio courses for the student to choose from.

A virtual structure differs from a conditional node in that it contains only a query. Instead of fixed text, the system displays the result of the query. So, in the previous example, if there were studio courses and the user responded to the question with a "yes," then the yes response might be implemented as a link to a virtual structure node with the query, `Display all courses which have studio_types`. The user would not see the statement of the query, just the results.

Conditional nodes and virtual structures add significant flexibility to hypermedia. They allow specific nodes to be responsive to changing conditions in other nodes of the hyperdocument. For instance, decision trees can be implemented using smart nodes by basing new decisions on nodes that contain the results of previous decisions.

The major surprise of this research was an important limitation of smart nodes. Suppose you had defined an inference computation for a specific node, embedded it in that node, and found that it worked fine. But now you wanted to apply the same computation to other nodes without explicitly entering the condition or query in each of the other nodes. More generally, suppose you wanted to apply the computation as an operation on an arbitrary list of nodes. This turned out to be a critical concern because it was important to be able to do this within the inferencing language itself.

Adding Smart Links

Smart *links* or *predicates* solved the limitation of smart *nodes*. Smart links are different from primitive links or defined link types. When a hypermedia system is designed, a set of link types is defined. For instance, in the academic advising application there might be links of type `proposed_courses` from a student's node to his or her chosen course nodes, and other links of type `prerequisites` from course nodes to other course nodes. A smart link would then be

a virtual link that was computed based on the definition of a predicate. For instance, a predicate might be defined as:

```
required_prerequisites = proposed_courses which have
prerequisites, with their prerequisites.
```

`Required_prerequisites` would not be a primitive defined link type, but a computation or an inference.

This is an example of a query using normal primitive links:

```
Display the proposed_courses for Sandra.
```

It would be evaluated by following the `proposed_courses` links from the student node `Sandra` and displaying the nodes reached:

```
*** PROPOSED_COURSES:
1. ENVD 2110  Architectural Studio
. . . .
```

This is an example of a query using smart links:

```
 Display the required_prerequisites for Sandra.
```

It would be evaluated by substituting the definition for the computed link type into the query and displaying the result:

```
***PROPOSED_COURSES:
1. ENVD 2110  Architectural Studio
     *** PREREQUISITES:
     1. ENVD 1000  Environmental Design Studio
     2. ENVD 1014  Intro to Environmental Design
     . . . .
```

The idea of substituting a definition for a term in a query is known as *macro expansion*. The definition of smart links as macros turns out to be an extremely powerful mechanism for the inferencing language. Because of the way the substitution is implemented, recursive definitions of smart links are possible. This allows simply stated queries to evaluate tree structures and easily display transitive closures, in both breadth-first and depth-first order -- an accomplishment not matched by relational query languages like SQL.

During our research, we further refined the process to distinguish between macros and predicates. A predicate is like a macro; however, when its results are displayed, they are labeled to appear as though the predicate were a primitive link type. This is critical for the user. Now when the user says,

```
Display the required_prerequisites for Sandra.
```

the user does not need to know that `required_prerequisites` is anything but an ordinary link type. The result is displayed like this:

```
*** REQUIRED_PREREQUISITES:
1. ENVD 2110  Architecture Studio
2. ENVD 1000  Environmental Design Studio
3. ENVD 1014  Intro to Environmental Design
. . . .
```

So now there are three kinds of links:

* Primitive links, which are the traditional link types of hypermedia.
* Macros, which add significant inferencing power.
* Predicates, which use the power of macros but hide the complexity from the user.

We had to define predicates like `required_prerequisites` and had to think about the differences between types, macros and predicates, but the user can use the computational power without knowing that no links exist between student nodes and their required prerequisites. The predicates look like simple links to the user. Therefore, they are called smart, computed or inferred links.

Smart links overcome the limitation of smart nodes. Because macros and predicates are syntactically equivalent to primitive link types, they can be bound to arbitrary nodes or lists of nodes as if they were actual links coming out of those nodes. Smart links turned out to be so powerful and flexible that we primarily developed the academic advising application with them. We incorporated smart nodes for only a few special situations.

The Academic Advising Application

The IBM HyperWin system provided not only the starting point for this research, but also the sample application for testing the results of our research: an academic advising system. The HyperWin version allowed a user to navigate through a hypermedia database of information about courses at Auburn University. This system asked the user about interests, courses already taken, etc., and responded to answers chosen by the user with appropriate further information.

To demonstrate the inference language, we created an application using information about the curriculum of the College of Environmental Design at the University of Colorado. This information included not only lists of offered courses, but other facts and rules used by the College's official student advisor. Courses were linked to their prerequisites and to their categories, such as which curriculum they belonged to and which elective breadth requirements they satisfied. Other, less formal factors were also included, like which courses were particularly labor intensive.

The centerpiece of this application was the definition of a predicate named `advice`. This predicate was built on a combination of several specific kinds of advice, which in turn used predicates to compute inferences across the hypermedia. The idea was that a student, Sandra, could enter her name, curriculum option, semester number, completed courses, current courses and proposed courses into the hypermedia system. By clicking on the Advice button, Sandra would initiate the query,

```
Display the advice to Sandra.
```

The query critiques Sandra's proposed list of courses. This is a typical result:

```
Here is some advice on your choice of courses:

The following courses each require a lot of work. It would
be wise not to take them in the same semester:
```

```
          ENVD 3220  Planning Studio 2
          MATH 1300  Calculus

The following courses are not designed for your curriculum
option:
          ENVD 3220  Planning Studio 2

You have not taken the listed prerequisites for the proposed
courses:
          ENVD 3220  Planning Studio 2

With your proposed courses you will not satisfy the
following elective breadth requirements:
          science
It would be wise to take a course in one of these areas
rather than the following proposed courses in elective areas
for which you have already satisfied the breadth
requirements.
          FINE 1012  Art History
```

The above is the actual system output for a sample student. Relatively intricate computations have been performed to check, count and list courses meeting or not meeting certain conditions. In particular, for instance, the advice about breadth requirements is only displayed if the proposed courses include an elective in an area that has already been satisfied and do not include one in an unsatisfied area. This kind of inferencing facilitates the offering of important information tailored to a particular user in a way that is impossible in a purely navigational hypermedia system. It begins to look like a rule-based expert system, but without many of the problems of such systems.

Developing an application of this level of complexity requires some system designing expertise. One needs to know how to represent the knowledge in hypermedia and how to build up a sequence of modular definitions. This is probably inevitable in any system. Once designed, however, the system is significantly easier to understand, modify, and extend than alternative implementations would be. While the result of the advice predicate looks like the output from a traditional expert system, the flexibility is still there to explore the underlying knowledge base by navigation. Alternatively, one can reformulate the major query or execute a series of simpler queries using components of the advice predicate.

Future Work

We developed the software discussed here in an object-oriented extension to Pascal in the MS-DOS operating system. We subsequently ported it to the Microsoft Windows 2.0 environment and enhanced it with a graphical user interface for defining predicates and queries in the language and for navigating through the hypermedia using smart nodes and links.

We are currently involved in embedding the language in a high-functionality design environment. (McCall, 1990) The specific application domain for this is lunar habitat design, a form of space-based architecture. (Stahl, 1993) This work entails three main aspects:

* Building a software environment to support designers, based on the kind of intelligent hypermedia discussed here.
* Extending the inferencing language to be multi-media, incorporating CAD-style vector graphics, bit mapped graphics, boolean conditionals, and numeric expressions, as well as text.
* Developing a hypermedia-based inheritance mechanism for type inheritance, virtual copying, perspectives, and versioning.

The language will continue to evolve in scope, power, and elegance as it is applied in new computational contexts and new application domains. We are interested in developing a language that can be written and read easily by people who are not experienced computer programmers. This will require considerable experience in observing the language in use in a variety of situations.

Conclusions

~~This research successfully extended hypermedia to handle inference. Some techniques were borrowed from artificial intelligence, and information retrieval has been integrated into hypermedia. This was done by developing an inference language that is fully integrated at the elemental level of nodes and links, that has broad power and generality, and that appears intuitive and English-like to the user. The paradigm of hypermedia was reconceptualized as intelligent hypermedia, built on the Inferencing Language.~~

The goal of our work has been to show that hypermedia has more computational potential than is generally thought. Hypermedia is often regarded merely as a user-friendly way of browsing "canned" information. We have endeavored to show that the link traversal mechanisms inherent in hypermedia also have the potential for knowledge-based computation. The key to exploiting this potential is twofold: 1) the creation of a powerful query language, and 2) the "burying" of queries at nodes and links to create virtual structures. The vast majority of our efforts in adding inference to hypermedia have concentrated on the former. Given a custom implementation of fine-granularity hypermedia which is object-oriented and designed with the language in mind, once a powerful retrieval language was developed only minor changes were needed to make the hypermedia perform full-blown inference.

Inference is defined as the combining of facts to derive new facts. In our approach to hypermedia, primitive nodes and links are combined by means of embedded predicates and queries written in the inference language to deduce new (virtual) nodes and links. The end effect for a user is the same as that created in expert systems by means of sophisticated inference engines, namely to infer logical or computational results. The *intelligent hypermedia* approach avoids the need for a separate inference module and spares the user the difficulties of formulating rule bases. The

computational mechanisms are handled by extending the normal navigational mechanisms of hypermedia.

The power of intelligent hypermedia was demonstrated in a suggestive way through the implementation of a realistic application, academic advising. Experience with this application and other test cases showed that the development of systems runs up against complications inherent in the domain. No matter how natural the language, an application may require the initial assistance of an experienced programmer or system analyst. However, for simpler tasks and, what is more important, for understanding, modifying, and extending existing applications, the inferencing language appears to be quite easy for users.

The inference language allows users to build a vocabulary of terms for their domain (or for their personal way of looking at things). This vocabulary is open-ended and can be modified or extended at any time. The naming of macros, predicates and queries allows the system designer or user to conceal technical complexities. Definitions can be built up step by step in a modular way to divide up complex concepts. In the end, statements can be formulated in a simple, natural way. The underlying details are always available to the interested user for exploring aspects of an overall problem or for tweaking a definition.

The research uncovered an unanticipated limitation of the original idea of smart nodes. We overcame this limitation with the notion of predicates. Instead of using approaches from procedural programming, we solved the problem in a way that is transparent to the user and results in intuitive results without the user having to be concerned with the use of variables. Using smart links implemented as predicates turned out to be a very powerful and useful notion. The power of this mechanism and of the inference language generally is discussed at length in (Stahl, 1991).

Academic advising is an example of a domain that could be modeled in a rule-based expert system or in navigational hypermedia. The use of intelligent hypermedia combines the computational power of the former with the flexibility of the latter.

Appendix: Abstract Syntax of the Inference Language

Following is the abstract syntax in BNF notation of the version of the inference language discussed in the paper. The options that existed in the original Mikroplis query language are underlined.

QUERY OPTIONS

Q : Query	Q::= display Art ***R Prp S which F***, if B, ***with their R***, else Q. | the Nth result of Q | Q and Q
S : Subject	S::= all items | ***all K*** | ***Z*** | Q | S and S
R : Relationship	R::= ***everything*** | ***T*** | converse T | R which are not self | ***the Nth R*** | R which F if B with their R, else R | Z | ***R Prp R*** | ***R and R*** | ***R as a macro*** | R as a predicate
F : Filter	F::= ***Eop C*** | Eop Q | ***have Qop R which F*** [, with them] | are Cop N |

e Cop N R which F [, with them] | F Lop F | ***F and which also F*** |
contain data type Top | contain no duplicates

MEDIA OPTIONS

C : Character — C::= ***String*** | {text of} Z | substring of C from N for N | C append C

N : Number — N::= Real | the count of results of the query: (Q) | N Nop N | Z

B : Boolean — B::= true | false | N is Cop N | C Eop C | Q Eop Q | not B | B Lop B | Pop Q | R of S which F

HYPERMEDIA OPTIONS

K : Kind — K::= node kind

T : Type — T::= ***link type*** | ***macro*** | predicate

Z : Node — Z::= ***C*** | C if B | Z if B | Q

L : Link — L::= ***Z linked to Z***

OPERATOR OPTIONS

Nop : Numerical — Nop::= plus | minus | times | divided by

Cop : Comparison — Cop::= more than | less than | the same as | at least | no more than | not the same as

Lop : Logical — Lop::= and | or | exclusive or | nand

Qop : Quantity — Qop::= no | all | (at least one) | most | several | a few

Pop : Proposition — Pop::= there are | there are not | there are several

Top : Type — Top::= numeric | non-numeric

Eop : Equality — Eop::= equal | are not equal to | do not contain | contain | are contained in | are not contained in

Prp : Preposition — Prp::= of | about | by | for | from | in | on | over | to | under

Art : Article — Art::= | a | all | an | that | the | those

References

Halasz, F.G. (1988). Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, Vol. 31, No. 7.

McCall, R. (1989). Mikroplis: A Hypertext System for Design. *Design Studies*, 10 (4), 228-238.

McCall, R., Bennett P., d'Oronzio P., Ostwald, J., Shipman, F., Wallace, N. (1990). Phidias: A PHI-based Design Environment Integrating CAD Graphics into Dynamic Hypertext. *Proceedings of the European Conference on Hypertext* (ECHT '90).

Peper, G., MacIntyre, C., Keenan, J. (1990). Hypertext: A New Approach for Implementing an Expert System. *Expert Systems ITL Conference 1989*.

Stahl, G. (1991). A Hypermedia Inference Language as an Alternative to Rule-Based Expert Systems. Tech Report CU-CS-557-91, Department of Computer Science, University of Colorado at Boulder.

Stahl, G. (1993). A Computational Medium for Supporting Interpretation in Design. *Journal of Architecture and Planning Research*, Special issue on Computational Representations of Knowledge, forthcoming in June, 1993.

Acknowledgments

The research reported here was supported in part by a grant from the Colorado Advanced Software Institute (CASI) for 1990-91, in collaboration with IBM. CASI is sponsored in part by the Colorado Advanced Technology Institute (CATI), an agency of the State of Colorado. CATI promotes advanced technology education and research at universities in Colorado for the purpose of economic development.