VALUATION LINKS:
FORMALLY EXTENDING THE
COMPUTATIONAL POWER OF HYPERTEXT

Michael Bieber
Tomas Isakowitz

Department of Information, Operations, and Management Sciences
Leonard N. Stern School of Business, New York University
44 West 4th Street, New York, NY 10012

VALUATION LINKS: FORMALLY EXTENDING THE COMPUTATIONAL POWER OF HYPERTEXT

by

Michael Bieber
Computer Science Department
Fulton 430
Boston College
Chestnut Hill, Massachusetts 02167-3808

and

Tomas Isakowitz
Information Systems Department
Leonard N. Stern School of Business
New York University
New York, New York 10003

June 1991

Center for Research on Information Systems
Information Systems Department
Leonard N. Stern School of Business
New York University

Working Paper Series

STERN IS-91-11

ABSTRACT

We view hypertext as an inherently dynamic concept to incorporate in the interface of dynamic information systems. What challenges does hypertext face in a constantly changing environment? In this paper, we discuss the benefits and the problems we face in our research into hypertext-oriented decision support systems. Then we focus on a new hypertext construct beneficial to this domain: *valuation links*. Valuation links support the dynamic spreading of computation via a well defined link traversal operation. We present two classes of such links: *static* and *dynamic*, and specify an algorithm for their traversal. We also show how these constructs can be used in sophisticated DSS environments.

KEYWORDS: Computer Interfaces, Decision Support Systems, Hypertext, Virtual Structures.

Valuation Links: Formally Extending the Computational Power of Hypertext

Many of today's information systems (decision support systems, executive information systems, database query systems, etc.) operate in dynamic environments where users need up-to-date information and this information changes rapidly. Hypertext browsing would enhance the functionality of these systems by giving users an intuitive way to access related details and annotations. Implementing this, however, presents several challenges to the traditional stable, "static" notion of hypertext where users are in control of changes, not the external environment.

In most of today's hypertext systems, link traversal affects the user's perception of the hyperdocument. Beyond procedural attachment for formatting, however, traversal generally has no impact upon node contents. Designers have tended to minimize interaction between node contents and the hyperdocument node and link structure in order to better formalize hypertext functionality and ease implementation. [CG88, SF89a, SF89b, Tom89]. Most interaction takes place during keyword/glossary search and other "querying" of node contents that result in the creation of a virtual link from its, e.g., keyword source to its definitional destination [Co87, YMvD85]. Even with query, however, we distinguish between content search and structure search—power is obtained from their combination [BK90, AK90, CM89]. We argue here that there is significant value in opening up hypertext to the richer, more dynamic environment that hypertext computation affords, where structural operations (such as link traversal) interact with hypertext content (such as generating and changing portions of text within a node, as we shall explore in sections 1 and 2 respectively).

Indeed, we believe that the power obtained from such functionality is especially important for general information systems that provide decision support and other intelligent interfaces. Furthermore, we believe this interaction should be formalized so its effects are controlled and do not present authors and users with unpleasant surprises when computation alters information unexpectedly. Current hypertext systems implement computation in an ad hoc manner (such as Guide's *command buttons* [GUI87]), generally requiring a knowledge of programming.

The purpose of this paper is to raise some of the issues we have faced in designing "dynamic" hypertext interfaces for decision support systems [BBK88, KPBB90].[1] In section 1 we begin by reviewing two mandatory characteristics of dynamic hypertext environments: virtual network structures and real time computation. Then in section 2 we describe problems we have come up against when modeling these in our research. In section 3 we focus on one particular challenge that dynamic information systems present to hypertext—updating the display contents of link markers so they remain accurate indicators of the links they represent. To do so we have formalized the concept of hypertext "valuation links" which update their source link marker when traversed. We demonstrate their usefulness in an information system that performs portfolio management. We conclude in section 4 by sharing our future research paths in modeling dynamic hypertext interfaces.

§1 Dynamic Hypertext

Two of the seven issues for the "next generation of hypermedia systems" that Frank Halasz puts forth in [Hal88] are virtual structures and computation over the knowledge base during link traversal.

In virtual structures, links are unresolved at the time a builder embeds their buttons

---

[1] Another example of a hypertext decision support system (DSS) can be found in [DeY89]. For a review of hypertext and DSS see [Mi89].

(link markers) in originating nodes for user selection. The builder must associate some routine with these buttons that resolves the identity of the link and destination node. One example is *Glasgow Online* [BMH88, BP89], which determines the actual destination of links pointing to train schedules according to the actual time of link traversal, in order to provide relevant information. As another example, Nguyen and Greenes [NG86] have developed a frame-based hypertext knowledge management system called EXPLORER-1 designed for the medical profession that supports various medical knowledge bases. Authors embed link marker "hot spots" in the frame declarations, which when selected, invoke either a procedure that resolves the identity of the destination frame based on a taxonomy of keywords contained by the frame or a procedure that can access a journal of the users transactions and obtain user input.

We need virtual linking ("computed linking" [Nie90]) because in a truly dynamic executive information system or decision support system, for example, it is quite possible that we cannot specify beforehand which links will be available from a link anchor.

Whereas resolving virtual links determines the links to traverse, "computation" involves creating new destination nodes. For example, exercising a "what-if" scenario generates analysis results to display. Expert system-style explanations and reports containing the most up-to-date information may need to be compiled in real time. Simply retrieving the current stock price for a display involves a call to a commercial database external to the hypertext system.

Several hypertext systems support some kind of programmed interface beyond procedural attachment (reformatting existing contents)—HyperCard [ACI] has buttons with underlying scripts, Guide [GUI87] has a "command button", KMS [AMY88] has "program links", NoteCards [Hal88] has access to an computational engine. Computation is implemented by invoking a procedure and results in the generation of a node, link or display value. One example is a KMS application that uses "computational hypertext" is Schnase and Leggett's Cassin's Sparrow energetics model [SL89]. This application recalculates values in a table and draws business graphs when the user selects a KMS program button.

We see the need for an enhancement to this degree of computation in a number of ways. First we may want to generate entire nodes on the fly. Second, changes to the hypertext network may originate outside the system and require automatic updates to the hyperdocument for display without user intervention [FS90, BIKM91]. Third, to track the interdependent effects of changes we need to maintain a dependency tree of all related system elements. We discuss this dynamic link dependency further in section 3.

Perhaps the greatest difference between most existing applications and the dynamic systems we are considering, however, is our underlying philosophy of hypertext as an inherently dynamic concept. In most existing systems the computational engine invoked, if any, is external to the hypertext system. In information systems with real time requirements, the hypertext computational engine should be an integral component of the system architecture. This is necessary because the vast majority of hypertext components are virtual entities, their identities, attributes and display values remaining unresolved until invoked. This presents several cognitive and technical challenges, which we describe in the next and in the final sections.

§2 Challenges to Hypertext in a Dynamic Environment

The word "dynamic" connotes "change" and change provides challenges to a generated hypertext network. Issues of change are compounded by the virtual structures and computation associated with a dynamic environment. Nodes and their contents may not exist (be neither resolved nor generated) until the link pointing to them is generated and

then traversed (the hypertext equivalent of "just in time" delivery). In this section we shall discuss several change-related issues and some solutions. In subsequent sections we shall concentrate on keeping displays up-to-date.

How does change affect a dynamically generated hypertext network? First of all, the user can never be sure that the link he traverses today will be available tomorrow. This is, of course, useful if you want the most up-to-date information, as will be the case in section 3. But it could lead to surprises for users who do not anticipate changes and have based user-declared links or comments on particular objects (e.g., the price of a stock) or reports containing these. For example, a decision maker may have based a (decision and) comment on the "final" sales figures for the quarter, not anticipating that the figures were in error and subsequently revised, invalidating both his decision and the comment. If the underlying parameters change then the user-declared link or comment may not be valid the next time the object is included in a generated report. The same goes for objects "pasted" into user-created documents when these documents are re-opened. This all calls for some type of version management, allowing the user to specify whether he or she wants to see up-to-date information or an older version. This is especially important for decision justification where an executive may need to recreate the information available at the time he made a particular decision or recommendation. In addition, being able to explain changes would be an especially helpful feature in a decision support or executive information system where the change, say, to a computed result may be caused indirectly by a change to an underlying data value or model parameter.

Another issue we face is determining when a user-declared link or comment is no longer valid, i.e., when its subject has changed so much that it is no longer accurate or the relation captured is no longer relevant. This is a problem in both static and dynamic environments. We may, however, be able to take advantage of the known structure of the application knowledge base to determine automatically when, at least a subset of these links and comments is no longer valid. If the user can declare some validity conditions based on parameter or data values in the underlying knowledge base then the hypertext engine should be able to check this before making the link or comment available. For example, the user may declare a comment or link to an expert system-generated plan of action, to be made available only when the price of a stock moves outside a certain, computed ranges. When this condition does exist, selecting the appropriate button will find this link or comment, otherwise it will not be presented as an option to the user.

§3 Valuation Links

In the rest of the paper we shall concentrate on a single theme within a dynamic hypertext environment—updating the display value of link markers. In particular we shall explore a new hypertext construct, *valuation links*, where the hypertext engine automatically updates the "source" link marker when underlying parameter values used in hypertext computation change. Recall the purpose of link markers—to serve as an indication of potential links to traverse. If updated information about the link to traverse becomes available, then to be accurate indicators, there will be times when marker values should change to reflect the current state of the system or the represented environment.

Example

A portfolio manager in an investment bank is in charge of several clients. He or she composes a personalized portfolio for each client conforming to some constraints. One of these constraints is the amount of risk involved in a given stock or bond. Some clients prefer low risk securities while others are more risk-inclined.

As part of his decision process the portfolio manager uses a hypertext system that supports valuation links. He describes a computational model that selects the securities for a particularly risk-averse client. The system receives input from

external sources on the risk levels of the securities in the client's portfolio. This input is channeled via a valuation link to the node containing the model used to configure the client's portfolio. As the risk level parameter changes in real-time inside the node due to external factors, the portfolio configuration model at the destination of the valuation link is dynamically re-evaluated.

This dynamic re-evaluation in turn might influence other nodes in the system. For example, the portfolio manager might keep a table in a hypertext node that specifies which stocks to buy and which stocks to sell. Next to each stock name the word "BUY" or "SELL" will appear as a valuation link marker associated with a destination node that is an analysis model. That model in turn might receive input from other nodes via other valuation links. We shall show later in this section that valuation links can propagate automatically in an iterative manner. Thus, a change in value in one node might trigger a sequence of re-evaluations which might change values in many nodes. For example, there may be several portfolios for different clients, or several "what-if" scenario portfolios containing a particular stock. When the price or other market factors change then the value and recommendation of all these related portfolios may need to be updated.

Valuation link markers can support standard hypertext functionality as well. For example, if the portfolio manager ever wishes to question a particular buy or sell decision he can do so by traversing a link from the valuation marker to a static (or generated) explanation of the underlying destination model. Thus valuation links can be treated referentially [Co87] when desired. The user can even perform "what-if" analysis by changing the model's equations or altering parameter values.

The previous example exhibits a seamless integration of different types of processes, data and action, all related by a sophisticated hypertext linkage mechanism. This environment provides an intuitive method for users to specify the composition of computations. The user is able to compose models on the fly by specifying that the output of one computation is to be used as the input for another one. It also permits interprocess communication to heterogeneous external systems. In UNIX the *pipe* paradigm is used for this effect. In our environment, valuation links take the place of pipes.

The PM System

We have implemented valuation links and the dependency tracking they require in the Maluar System [Isa91]. Written in LISP, Maluar makes LISP available to all nodes. Maluar provides a basic set of tools that can be used to create more elaborate hypertext environments. One such environment is the PM system [BIKM91], which captures the decisions a portfolio manager (PM) in an investment bank. The PM makes portfolio decisions through some combination of doing analysis, and using information from the bank's industry-specialist analysts and from external systems. Both analysts and PMs have the same sources of information at their disposal. If they were to use the same models and had the same assumptions embedded in their evaluation heuristics, they would arrive at similar recommendations. In reality, however, there is asymmetry in information, decision criteria and assumptions. Such asymmetry gives rise to potential for conflicts. A portfolio manager may come up with a recommendation of "buy" based on the analysis that he makes and the rules that he has for making such decisions. On the same item an analyst may come up with a contradictory recommendation. The PM system uses the Maluar system's dependency tracking and automatically updating valuation links to not only support the processes that converge to a decision on a given stock item, but also to be able to detect discrepancies between PMs and analysts, and act upon these. The system contains both nodes representing models that embody heuristics for portfolio composition and *conflict detection* nodes which continuously and automatically monitor for discrepancies among nodes used by analysts and PMs.

Figures 1 and 2 present an example of the system in use. Linda Goodman is an analyst and Don Burnt is a PM. Both issue recommendations upon the stocks of Prudential using the following information: number of shares, projected dividends and dividends per share (which is computed dynamically). Each, however, uses different heuristics which are embodied in a node which dynamically evaluates a recommendation: "buy", "sell" or "hold". In each figure the PM system determines that there is a conflict.

The first case of conflict is demonstrated in Figure 1, in which links between nodes are depicted as arrows indicating the direction of information flow in the link. The information that Don has helps him infer that the stock should be "held". Linda's information, however, leads her to infer a "buy" action. Backtracking from Linda's recommendation leads Don to the information asymmetry, that is the node that contains "projected dividends". Once this is corrected, the other valuation link markers are updated automatically.

The second case of conflict is demonstrated in Figure 2. Here Don's heuristic helps him infer that the stock should be "sold". Linda's heuristic leads her to infer that the action to be taken is "hold". In this case, following the links back from Linda's recommendation leads Don to the conclusion that Linda uses a different heuristic, and this can help him judge future recommendations from Linda.

A Formal Description of Valuation Links

We are proposing a new hypertext construct, *valuation links*, and have sought to motivate them through the examples above. Valuation links have their source link marker in a visible node and have a computational program as a destination node. The effect of traversing a valuation link is to update the display value of the source link marker with the result of the destination node's computation. While such behavior is mentioned in the Trellis hypertext reference model [FS90] and admittedly could be achieved by many of the *ad hoc* computational mechanisms described in §1 (e.g., Guide *command buttons*), as yet we have seen no formal link taxonomy [DeR89, Nie90] with this functionality. This is what we present here.

We identify two kinds of valuation links:

static valuation links, the computation associated with the link is performed upon demand by explicit user request, and

dynamic valuation links, the link is automatically activated whenever its destination is updated.

Previously we have given several examples of dynamic valuation links. Static valuation links are also useful. Suppose a manager creates a report where there is a static valuation link representing projected sales. The static link is connected to a projection model. As new sales data is fed into the system, the projection figures change. The manager, however only re-evaluates his figures when he determines a need to do so. In the meantime he does not want the report to change.

In order to traverse these links an algorithm is needed to help keep the up with the potential chain reaction of re-evaluations which might be triggered by a single link activation. The following algorithm manages evaluation of *value regions* (i.e., link anchors) associated with valuation links and programs that return values. A value is associated with each *value region* and with each *valuation link*. The strategy we adopt here forces each *value region* to always keep the current value of the computation it represents stored in a variable. Every time a node content change is performed, the affected links are computed and propagation takes place. There are three functions:

```
function value-traverse(value-link)
   begin
      retrieve_source_region(value-link, source-region);
      paste value of value-link in source-region;
      if source-region is a value-region
         then evaluate(source-region);
   end;

function update-value(value-link, val)
   begin
      set value of value-link to val;
      if value-link is a hot-link
         then value-traverse(value-link);
   end;

function evaluate-region(value-region)
   begin
      val := result of execution of value-region;
      for each value-link ending in value-region
         update-value(value-link, val);
   end;
```

The difference between dynamic and static valuation links is in the *update-value* function. In the case that the link is dynamic, after its value is updated, the link is traversed. This in turn will cause the value region at the source of the link to be evaluated, the display value of incoming value links to be updated, and so on.

From these three recursive functions and their types of dynamic interaction they support, it should be clear that valuation links are a general and powerful tool for dynamic information systems.

§4 Discussion

This paper concerns itself with incorporating computational power into hypertext in a well defined manner. Although the idea of computational hypertext is not new, many of the approaches taken elsewhere [GUI87, AMY88, ACI—HyperCard] are not rigorous in their specification and abandon the *link traversal* metaphor to some extent. The *valuation links* introduced here provide rich computational power while staying within the framework of hypertext. In doing so we revisited the meaning of *link traversal* to make it dependent upon the type of link being traversed, but beyond the notion of procedural attachment. We do this more in the spirit of the corresponding *access* operation for nodes. It has been recognized that the exact meaning of node access varies depending upon the node type: text nodes are presented in a window; video nodes are projected; audio nodes are played; program nodes are evaluated. Similarly, traversing *referential links* is interpreted as access to the destination node, whereas traversal of, e.g., *note* links in Guide [GUI87] relates to pop-up a window. Traversal of *valuation links* entails the transfer of values. What is achieved by using this paradigm is the incorporation of powerful computational mechanisms while remaining within the *browsing* paradigm that characterizes hypertext. This computational power is key to the acceptance of hypertext as a decision support tool and paves the way for a larger user audience.

Our intended research takes us deeper in the domain of formal models that explicate the nature of computation (e.g., [Bi90]). One route we may take is incorporating type dependent link traversal into a formal model along the lines of the Trellis model [SF89a]. In any event we shall contine to exploit the the power of computational

hypertext to develop sophisticated but simple-to-use information systems for decision support.

Acknowledgements
The authors want to thank Raghav K. Madhavan and P. R. Balasubramanian for their technical support for this paper.

References

[ACI] Apple Computer, Inc., 20525 Mariani Avenue, Cupertino, CA 95014.

[AK90] F. Afrati and D. C. Koutras, "A Hypertext Model Supporting Query Mechanisms", in *Proceedings of the European Conference on Hypertext*, 1990.

[AMY88] Robert M. Akscyn, D. L. McCracken, and E. A. Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations," *Communications of the ACM* 31(7):820-835, 1988.

[BBK88] Hemant Bhargava, Michael Bieber, and Steven O. Kimbrough, "Oona, Max and the WYWWYWI Principle: Generalized Hypertext and Model Management in a Symbolic Programming Environment, " in Janice I. DeGross and Margarethe H. Olson, editors, *Proceedings of the Ninth ICIS*, pages 179-192, 1988.

[Bi90] Michael Bieber, Generalized Hypertext in a Knowledge-based DSS Shell Environment, Ph. D. Dissertation, Decision Sciences Department, University of Pennsylvania, 1990.

[BIKM91] P. R. Balasubramanian, Tomás Isakowitz, Rob Kauffman, and Raghav K. Madhavan, "Hypertext in Risk Management," Technical Report, NYU, 1991.

[BK90] Catriel Beeri and Yoram Kornatzky, "A Logical Query Language for Hypertext Systems," in *Proceedings of the European Conference on Hypertext*, 1990.

[BMH88] P. Baird, N. Mac Morrow, and L. Hardman, "Cognitive Aspects of Constructing Non-linear Documents: HyperCard and Glasgow Online," in *Proceedings Online Information '88*, pages 207-218, December 1988.

[BP89] P. Baird and M. Percival, "Glasgow Online: Database Development Using HyperCard," in R. Mc Aleese, editor, *HyperText Theory into Practice*, 1989, pages 75-92.

[CG88] B. Campbell and J. M. Goodman, "HAM: A General Purpose Hypertext Abstract Machine," *Communications of the ACM*, July 1988.

[CM89] M. P. Consens and A. O. Mendelzon, "Expressing structural Hypertext Queries in GraphLog," in *Hypertext '89 Proceedings*, pages 249-258.

[Co87] Jeffrey Conklin, "Hypertext: a Survey and Introduction," *IEEE Computer*, V20:9, 1987, 17-41.

[CPWR86] D. Canter, J. Powell, J. Wishart, and C. Roderick, "User Navigation in Complex Data Base Systems," *Behaviour and Information Technology*, 5(3):249-257, 1986.

[DeR89] Steven J. DeRose, "Expanding the Notion of Links," in *Hypertext '89 Proceedings*, pages 249-258, 1989.

[DeY89] Laura DeYoung, "Hypertext Challenges in the Auditing Domain," in *Hypertext '89 Proceedings*, pages 169-180, 1989.

[FS90] Richard Furuta and P. David Stotts, "The Trellis Hypertext Reference Model," *Proceedings of the Hypertext Standardization Workshop*, NIST Special Publication SP500-178, 83-94.

[GUI87] Guide User's Manual, Owl International Inc., 1428 NE 21 St., Bellevue, WA 98007, 1987.

[Hal88] Frank G. Halasz, "Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems," in *Communications of the ACM*, 31(7):836-852, 1988.

[Isa91] Tomás Isakowitz, "MALUAR - A Computational Hypertext Environment," Technical Report, NYU, 1991.

[KM89] M. R. Kibby and J. T. Mayes, "Towards Intelligent Hypertext," in R. Mc Aleese, editor, in Hypertext Theory into Practice, pages 164-172, 1989.

[KPBB90] Steven O. Kimbrough, Clark Prichett, Michael Bieber and Hemant Bhargava, "The Coast Guard's KSS Project", *Interfaces*, V20:6, November/December 1990, 5-16.

[Mi89] Robert Minch, "Application and Research Areas for Hypertext in Decision Support Systems," *Journal of Management Information Systems*, V6:3, Winter 1989-90, 119-138.

[NG86] L. T. Nguyen and Robert A. Greenes, "A Framework for the Use of Computed Links in the EXPLORER-1 Knowledge Management System," in *MEDINFO 86, IFIP-IMIA*, R. Salamon, B. Blum and M. Jørgensen (eds), North-Holland: Elsevier Science Publishers B.V., 1986, pages 891-894.

[Nie90] Jakob Nielsen, Hypertext & Hypermedia, Academic Press, 1990.

[SF89a] P. David Stotts and Richard Furuta, "Petri-Net Based Hypertext: Document Structure with Browsing Semantics," *ACM Transactions on Information Systems*, 7(1), January 1989.

[SF89b] P. David Stotts and Richard Furuta, "Programmable Browsing Semantics in Trellis," in *Hypertext '89 Proceedings*, pages 27-42, 1989.

[SL89] John L. Schnase and John J. Leggett, "Computational Hypertext in Biological Modelling," in *Proceedings of the 1989 Hypertext Conference*, Pittsburgh, PA, November 1989, pages 181-198.

[Tom89] Frank Wm. Tompa, "A Data Model for Flexible Hypertext Database Systems," *ACM Transactions on Information Systems*, 7(1), January 1989.

[YMvD85] Nicole Yankelovich, Norman Meyrowitz, and Andries van Dam, "Reading and Writing the Electronic Book," *IEEE Computer*, October, 1985.