Expert Systems With Applications, Vol. 5, pp. 369–375, 1992
Printed in the USA.

0957-4174/92 $5.00 + .00

Knowledge Acquisition Using Hypertext

MIKAEL SNAPRUD

Technische Universität Wien, Vienna, Austria

HERMANN KAINDL

Siemens AG Österreich, Vienna, Austria

Abstract—*This paper addresses the issue of supporting knowledge acquisition using* hypertext. *We propose a way of tightly integrating hypertext and* structured object representation, *using Artificial Intelligence (AI) frames for the basic representation of hypertext nodes. Epistemologically, a* dual *view of the resulting space is of interest. One view is that of hypertext which emphasizes nodes containing text, including formal knowledge representation. The other view focuses on objects with certain relationships, which define a* semantic network. *Both in hypertext and in semantic networks the relations between chunks of knowledge are explicitly represented by links. However, in today's hypertext systems a node typically contains just informal text and references to other nodes. Our approach additionally facilitates the explicit representation of structure "inside" hypertext nodes using* partitions. *We show the usefulness of such a tight integration for knowledge acquisition, providing several features useful for supporting it based on a level of basic hypertext functionality. In particular, we sketch a method for doing knowledge acquisition in such an environment. Hypertext is used as a mediating "semiformal" representation, which allows experts to directly represent knowledge without the immediate support of knowledge engineers. These help then to make this knowledge operational, supported by the system's facility to provide templates as well as their links to the semiformal representation. As an example of our results of using this method of knowledge acquisition, we illustrate the strategic knowledge in our application domain. More generally, our approach supports important aspects of (software) engineering knowledge-based systems and their maintenance. Also their user interface can be improved this way.*

1. INTRODUCTION

THERE IS A STRIKING SIMILARITY between *hypertext* and *semantic networks* due to the common elements of *nodes* and *links*. Both approaches are characterized by explicit relations between chunks of knowledge (represented informally or formally). In addition, the links represent important knowledge themselves. "Typed" links even emphasize this. Hence, we can support the view that in hypertext this amounts to a "semiformal" representation of knowledge. We prefer this notion (see also Jordan, Russell, Jensen, & Rogers, 1989) to "semistructured" as used in Malone, Grant, Turbak, Brobst, & Cohen (1987), since a hypertext may well be completely structured, while the textual content of nodes is typically informal.

In addition, we propose to allow for the additional option of explicit representation of structure "inside" hypertext nodes using *partitions*. *Frames* facilitate such a representation via *slots*, much as they do for representing the internal structure of nodes in a semantic network. While we believe in the importance of questions of what these representation schemes exactly mean in a formal sense (see for instance Hayes, 1979), we focus in this paper on integration with an informal way of representing "knowledge."

We use a tight integration of hypertext with frame-based representation for supporting knowledge acquisition. The semiformal way of knowledge representation in hypertext may serve as a mediating representation between informal and formal to be used for coping with the representation mismatch, which has been identified as a major issue of knowledge acqui-

---

Revised version of Snaprud, M., & Kraindl, H., Knowledge acquisition using Hypertext, pp. 781–788, from Liebowitz: *Expert Systems World Congress Proceedings*, copyright 1991, with permission from Pergamon Press Ltd., Headington Hill Hall, Oxford, OX3 0BW, United Kingdom.

M. Snaprud's work is partially supported by the Austrian Fonds zur Förderung der wissenschaftlichen Forschung (Project No. P7857-TEC).

Requests for reprints should be sent to Mikael Snaprud, Technische Universität Wien, Institute of Machine and Process Automation, Gußhausstraße 27–29, A-1040 Vienna, Austria.

sition by Gruber (1989). This way, experts can become more independent of knowledge engineers, since they can represent their knowledge informally, first as linear text and formalize it step-by-step introducing hypertext links and partitions of text. Only then, together with a knowledge engineer, this knowledge is made operational for the machine. We have experimented with this approach in the domain of modeling and identification of systems. The objective of *system identification* (Ljung, 1986) is to derive a dynamic model of a technical system (e.g., a chemical plant). This model can be used to control or simulate the system.

First, we outline a tight integration between hypertext and *structured object representation*, using Artificial Intelligence (AI) *frames* for the basic representation of hypertext nodes. Thereafter, we show its usefulness for knowledge acquisition. In particular, we sketch a method for doing knowledge acquisition in such an environment. Finally, we discuss the positive effects of this approach on the (software) design and maintenance of knowledge-based systems.

2. OUTLINE OF THE TIGHT INTEGRATION

We have implemented our approach by use of a hybrid tool (KEE). However, only *frame-based representation* is necessary for implementing the key features of this approach. Since the term "frame" is also often used in the context of hypertext, we note that it is meant in this paper in the sense of Minsky (1975) and Hayes (1979). Our representation of hypertext is based on such frames, much the same way as other (formal) knowledge representation paradigms are often based on it, e.g., rules, methods, and active values.

Intentionally this representation is fairly straightforward, since we wanted to quickly get a basis for experiments, and we think it is the natural way of realizing this approach in such an environment. Moreover, we want to show others how easily they could use it, too.

As a more distinctive feature, our approach facilitates the explicit *partitioning* of the hypertext nodes, which is implemented using slots. The records implementing the modules in the *Document Examiner* (Walker, 1987) are similarly composed of so-called fields. While these are mainly used to represent standard information like name or version number to be used by the editor and by other supporting software, our emphasis is to support the user in partitioning the textual content.

2.1. A Dual View of the Overall Representation

First of all, we think that an epistemological view of the approach may be useful. Figure 1 shows an example knowledge base which is organized as a "hypertext space". It is divided into two disjoint "halfspaces". One contains hypertext nodes with informal knowledge representation, the other operational objects associated with them via hypertext links. The example is the prototypical implementation of hypertext itself. So, the latter represent the operational knowledge of this implementation, while the former document it. This example suggests that the hierarchical structure of the documentation is nearly the same as that of the operational part. Although this is by no means necessary, the approach to knowledge acquisition described below results in such a correspondence.

The figure only shows the hierarchical links in the inheritance lattice. Of course, there are also other links between nodes containing text, and the objects containing machine interpretable representations may be related as necessary. The latter represent relationships between the two halfspaces. From the hypertext view, these are just special "typed" links. All nodes are either in one or in the other halfspace, more precisely, they are defined to be. For supporting convenient navigation—especially between the two halfspaces—we use a kind of bidirectional link. The explicitly given link in the text of node F points *from* F to the referenced node T. Its inverse points from T to F, and it is maintained automatically.

Especially for use in knowledge acquisition, we found it useful to allow also for the possibility of links *into* nodes. In our representation (as described below), this is conveniently implemented via links to *slots*.

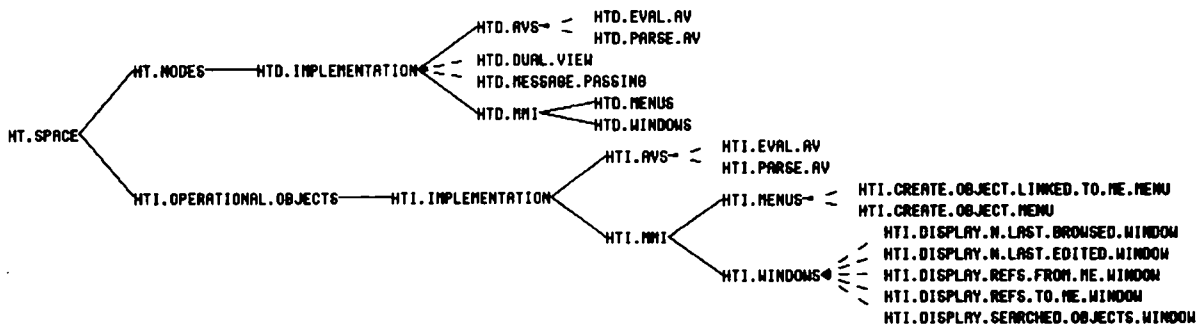

FIGURE 1. Taxonomy of a hypertext space.

This is the view of hypertext, the *dual* one is that of a semantic network. The former emphasizes nodes containing text, which includes "special" nodes with formal knowledge representation. The latter view focuses on objects with certain relationships, of which links between text are just a special case.

2.2. Basic Representation of Hypertext Nodes

The basic element in our approach is an (AI) frame. While this is not completely novel (see Carlson & Ram, 1990; Koh, Loo, & Chua, 1990; Hofmann, Schreiweis, & Langendörfer, 1990), none of the descriptions in the literature is very explicit (with the exception of Fikes, 1988, where a different approach than ours is described). Moreover, while frames are used for higher-level structuring in the approach presented in Koh et al. (1990), the hypertext nodes are not represented as frames there.

In our approach, a hypertext node is represented as an (AI) frame. Text is stored in a slot, which is originally inherited from HT.NODES. This text contains links to other nodes (in the hypertext view). These are represented in the same way as any reference to another frame in the basic system (KEE). However, the links are indirectly stored within the frame, i.e., the real reference is stored in a specific slot, and only this is referred to from the text. The main reasons for chosing this internal representation are related to the maintenance of the objects and their names, in particular to renaming and deletion.

The *partitions* of a hypertext node are also realized using slots. This approach is especially useful for our current main goal, using hypertext as intermediary representation in the process of knowledge acquisition for a frame-based system.

Although it would be easy to provide each frame in the system with a slot for representing text (via inheritance), we decided not to have such text in the frames containing formal knowledge representation. It is only possible to have hypertext links with such frames. The main reason is that we want to have a clear separation between the two halfspaces. Generally, such a representation using frames offers all the possibilities to easily implement more sophisticated hypertext concepts, for instance composite nodes.

One of the most important issues for selecting and evaluating a representation is its use. In particular, our chosen representation facilitates the basic operations for entering, editing, and browsing hypertext nodes. We designed and implemented a browser which completely hides the internal representation from the user. The links are highlighted and mouse-sensitive, and clicking them leads to browsing, editing, or even other actions on the referenced node or partition. It also turned out to be useful to have the possibility of performing operations on several nodes at once.

If references to nonexisting frames are added by the user (as checked by the used tool), then *templates* are created by means of dummy frames to be filled later. The inverse link, however, is entered immediately. This creation is supported by menus, which allow the user to select different kinds of templates. Additionally, this selection is supported by the possibility of clicking on the lattice, creating templates related to existing objects. By entering the new node into the taxonomy represented by the lattice, it can inherit text as well as links (cf. IDE's "template cards" (Jordan et al., 1989).

Many of today's hypertext systems support the activation of procedures in the course of traversing a link. Our approach facilitates object-oriented programming by procedural attachment to hypertext nodes. *Message passing* is possible from the text in the browser, in effect making it "active text". For instance, text could be returned which reflects the current status of some part of the knowledge base.

2.3. Support for Navigation

While there are many issues of designing hypertext in order to facilitate *navigation* (see for instance (Nielsen, 1990)), our focus was first of all on providing simple but effective support for design tasks, in particular the design of knowledge-based systems. Since here active search in the hypertext was initially not of primary importance, we did not support this yet by a query language. However, we found it useful to have the possibility of string searching. The scope of such a search can be specified within parts of the hierarchy of nodes or by other criteria (for instance selecting all the nodes linked to a given one). The result of such a search can be used to narrow the scope of a subsequent search, in effect focussing the search process step-by-step. Moreover, there is support for changing strings and locating node names. Another possibility of defining the scope of searching is related to the specification of partitions to be searched or not.

As indicated above, there are several operations possible on a selected object in the user interface. One simply leads to the display of the referenced node, another to editing it. In addition, there is the option to display only the "refs from me", i.e., a list of the explicit links from the referenced node without the text they are embedded in. Analogously, it is possible to display the "refs to me" of a node, utilizing the inverse links. Browsing of nodes in the operational halfspace always leads to a display of "refs to me" since there is no other "text" to show. This way, a convenient possibility is given to reference the nodes describing such an object.

Two *history lists* keep track of browsed and edited nodes, respectively. (The earlier systems usually maintained only one.) These lists facilitate the usual operation of *backtrack*, as well as the more advanced operation called *backjump* in the hypertext information

system HIS (Kaindl & Ziegeler, 1992) (in analogy to the procedure with this name proposed by Gaschnig, 1979, for *constraint satisfaction*, and as used, e.g., in Ziegeler and Kaindl, 1991). The latter operation allows jumping back directly to any node on the currently visited path in hyperspace. Both these lists can be used for editing, browsing, and displaying the "refs to me" and the "refs from me" of the stored entries.

While we unfortunately have different interfaces for editing and browsing, we can mix these actions arbitrarily, e.g., it is immediately possible to edit a node referenced from the browser, and even the currently browsed node. (Several hypertext systems today have different programs for these operations, which poses certain restrictions on their use.) We think that this freedom of use is important for supporting *design* activities.

3. KNOWLEDGE ENGINEERING

Such marked design phases like that in conventional software engineering may neither be useful nor necessary in the process of building a knowledge-based (expert) system. However, the immediate representation of just-acquired knowledge in a fully formalized way sometimes leads to epistemologically bad representations. Hence, whenever it is not completely clear how to represent something appropriately, structured text may first of all serve as a means for semiformal representation. Once having represented this chunk of knowledge formally, it is immediately connected to the text in our approach. This process leads then directly to a useful documentation of the knowledge base.

Although not every detail of this documentation will be of interest to the user of the resulting knowledge-based system, selected parts of it may be directly used for explanations. Also without tight integration, hypertext can be used for improving the user interface of a knowledge-based system in case certain explanations are needed (see Rochowiak, Ragsdell, & Wurzelbacher, 1989 and Linster and Gaines (1990)). In addition, our tight integration can utilize "active text" for adapting variable parts to the current situation. Its behavior can also be made depending on the (type of the) node from where it is activated, and it can be inherited. This feature can be useful both for documentation and for help text. Moreover, the references to the literature and the glossary of domain terms we created in hypertext during knowledge acquisition can be used also by the end user.

3.1. Knowledge Elicitation and Acquisition

Most of the more concrete work on AI and hypertext dealt with the support of knowledge acquisition, since hypertext appears to provide a useful means of human–computer interaction for this purpose. The first approaches were based on existing hypertext tools such as *NoteCards* (Gaines & Sharp, 1987). A reason for using hypertext before writing for instance rules is to avoid deciding on a *tool* before knowledge acquisition (Rochowiak, 1990). While this issue is less important when using a hybrid tool, still the decision on a *formalism* before knowledge acquisition is to be avoided (Wells, 1989).

In our opinion, it is more promising to integrate hypertext in a frame-based environment. Again, an important reason is the support of knowledge acquisition, for instance by a hypertext system implemented in *Smalltalk* (Hofmann et al., 1990). In addition, there are approaches of loosely combining a hypertext system with a knowledge-based system or environment (for instance *HyperCard* and CLIPS (Rochowiak et al., 1989). HyperCard has also been combined with BABYLON and moreover with the knowledge acquisition tool KSS0 into one environment named *Hyper-KSE* (Linster & Gaines, 1990). A comparable integration is presented in Gaines, Rappaport, & Shaw (1989).

In fact, it looks very appealing to have the possibility to represent the knowledge informally first, since the experts provide it also in such a form. This way the very difficult process of modeling can be done more consciously, based on the explicit representation as hypertext. Still, much work is needed to find appropriate methods for actually doing it right.

Similar to a "rhetoric" for authoring hypertext as intended for readers (Landow, 1989), rules for designers using hypertext will have to be formulated. In the system gIBIS (Conklin & Begeman, 1988) special links are used for supporting cooperative work. We have experimented with *special nodes* for use in knowledge acquisition, such as WHY, HOW, TO.DO, DIARY, etc. In this respect, we make heavy use of the "refs to me". For instance, the TO.DO node serves as an "agenda". We can write the reference to this node into some of the hypertext nodes. Selecting TO.DO for the "refs to me" lists all these nodes. Analogous treatment of WHY leads to the functionality of "question cards" in Hofmann et al. (1990). However, our implementation appears to be better due to the use of the inverse links.

Now let us briefly sketch how we are actually doing knowledge acquisition in this environment. First, we create and edit hypertext nodes, describing the ideas and concepts to be represented and the issues involved. For instance, also references to the literature and definitions of terms resulting in a kind of glossary can be represented at this stage. Whenever, in the course of editing a node, another not yet existing node is referenced, the system detects this and helps to create a template. When appropriate, these templates are filled with informal text. Meanwhile some related idea may have come to mind which is immediately described in a hypertext node, etc.

From this procedure, many useful ideas and concepts should arise. However, they still may be unordered or only slightly ordered. Hence, there should be a phase of ordering the concepts, which may lead to a *taxonomy.*

Taxonomies play a major role in object-oriented programming and for structuring knowledge bases. Hierarchies in general are considered a backbone of hypertext in order to facilitate navigation. Organizing a hypertext "knowledge base" around a taxonomy of the stored concepts has proven useful in the hypertext information system HIS (Kaindl & Ziegeler, 1992). Hence, we think that in an integrated environment, hierarchical structuring is at least as useful. While it is possible to define special types of hierarchical links for hypertext nodes, we chose to directly use the *subclass/superclass* and *instance* relationships as supported by the used tool for this purpose. This way the built-in mechanism for inheritance can be used for free.

When the internal structure of a concept is becoming more and more clear, it can be conveniently represented first using partitions in the corresponding hypertext node. When the concrete formal representation is to be done, this structure can be transferred to the corresponding frames in the other halfspace (supported by the system). This leads to templates of slots, which are filled then with operational knowledge.

Generally, whenever a part of the hypertext description appears to be well structured, a *transfer* to the operational halfspace can be initiated. Currently, the user can select a subtree in the hierarchy for this transfer. The system creates then a template for this subtree in the halfspace of the knowledge base which will contain the operational objects. Moreover, it establishes links between the original hypertext nodes and their operational counterparts. Such a template contains (part of) the structure of these nodes, i.e., their mutual relationships as well as their internal partitioning. This structure serves as a basis for the completely formal (operational) representation.

This method of dealing with templates for groups of nodes has to be distinguished from the one described in Jordan et al. (1989), which allows a hypertext user to generate several nodes by one action, based on an intensional description of the structure. Our templates resulting from a transfer represent the structure which was extensionally represented in the semiformal hypertext description. The goal in our approach is to filter this structure from the informal descriptions in order to get a basis for the formal knowledge representation, while keeping the descriptions associated by links.

An example of our results using this method of knowledge acquisition is depicted in Figure 2. It shows the data flow and control flow of system identification. This version of the so-called identification loop is more elaborated than any such representation we found in the corresponding literature. In particular, we emphasize the explicit distinction between control flow and data flow using typed links in our hypertext representation. According to our knowledge acquisition methodology we developed it by first identifying steps (shown in boxes) and data/knowledge units (shown in circles), and representing them using hypertext nodes. Then they were described using informal text which originates partially from books and partially from our own considerations and discussions with an expert. Relations between these nodes were represented explicitly by use of typed hypertext links, and the internal structure of the textual descriptions was represented by partitions. The external and the internal structure were mapped then to the operational halfspace.

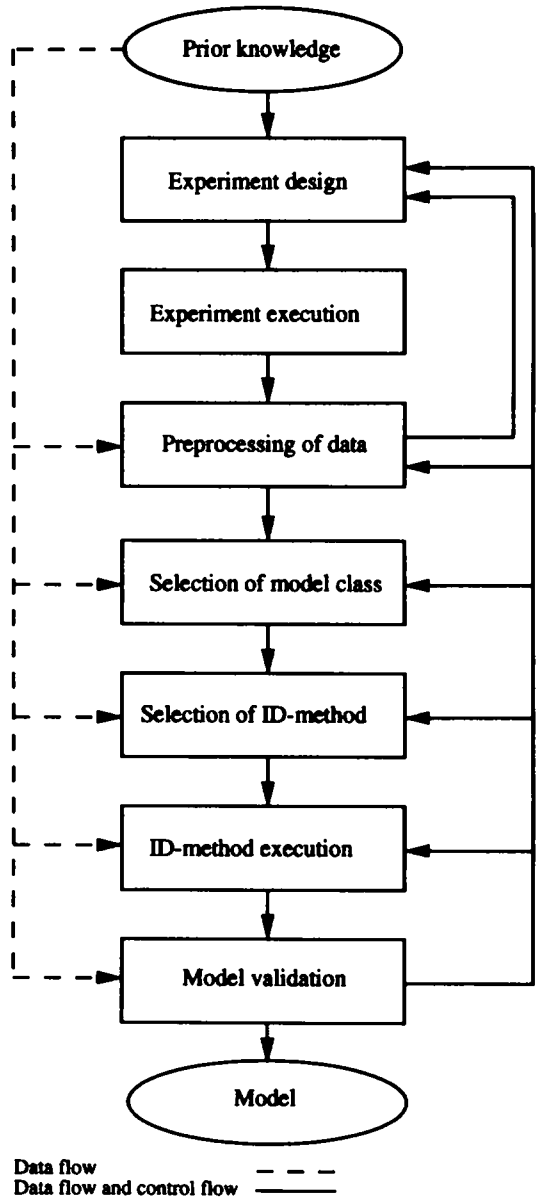

FIGURE 2. Data flow and control flow of system identification.

The resulting frames and slots can be viewed as a declarative representation of *strategic* (control)

knowledge. An interpreter uses it for driving the identification task. Uniformly, the frames representing a step have slots containing rule classes to be activated and messages to be passed. The interpreter just initiates these in the given order, taking into account also some given conditions.

An interesting point is that on the global level there are several options to return to a previous step. (Hence, this can be viewed as an *indeterministic algorithm.*) In this domain it is a delicate problem to determine the step to return to. However, the following metaprinciple representing *abstract control knowledge* can help. Whenever some subsequent step does not achieve satisfactory results, it must propose some change in the current assumptions of the overall procedure. Since each step has some input associated with it, the step to return to is identified as the nearest preceding one whose input data are affected by this change.

Based on the central philosophy of tightly integrating hypertext and knowledge-based systems, we can approach a powerful methodology for knowledge acquisition. The enhancements of special nodes and links for this purpose can be seen as steps towards an advanced knowledge acquisition tool.

In effect, hypertext serves as a mediating representation between the informal one given by the expert and the formal representation intelligible for the machine. This way, it is useful for coping with the representation mismatch (Gruber, 1989). In particular, *modeling* is made easier since an explicit (semiformal) representation of the knowledge is already there, facilitating *operationalisation.*

3.2. Support for (Software) Engineering the Knowledge-Based System

Hypertext also has high potential to support software engineering (see Bigelow (1988) and Garg & Scacchi (1989)). For the approach of additionally using AI methods, the notion HAISE (hypertext and artificial intelligence techniques to support software engineering) was coined in Carando (1989). Since knowledge-based systems are typically software, the question is whether and how these approaches can also support the development of such systems.

In fact, the process of knowledge acquisition sketched above adds a flavor of top-down design to the more usual bottom-up procedure of building knowledge-based systems. However, it is not in accordance with the pure waterfall model of traditional software design. We believe that there should be a balance between top-down and bottom-up processing, depending on the degree of prior understanding of the domain concepts. Most of the advantages of using hypertext for developing traditional software carry over to the development of knowledge-based systems. These are primarily related to project management and documentation. While for the support of project management in our environment additional effort will be necessary, integrated documentation is gained more or less for free.

This implies advantages for maintenance. Since the knowledge base and its documentation is available in one system, consistency can be achieved more easily. Using the feature of "active text" also helps keeping the documentation up to date. Moreover, the structure of the documentation can be more or less the same as that of the operational knowledge base (see for instance Fig. 1). Especially for larger knowledge bases and for the case that the maintenance personnel is different from the developers, there is another advantage. Entry to the details of the operational knowledge representation can be through its hypertext description.

Generally, an important issue of software management is how software can be reused. Although especially from an economical point of view this would be highly desirable, both in theory and in practice there are major obstacles. In particular, the same applies to knowledge-based systems viewed as software. While we do not believe that our approach can solve this issue completely, we think it may help to improve the situation. When a knowledge base is well documented it is more likely that at least pieces of it can be used again.

4. CONCLUSION

Combining hypertext with structured object representation appears to be very useful. We argued for a tight integration such as the one outlined in this paper. Primarily, support for knowledge acquisition suggests itself. According to our experience, however, it is very important to efficiently manipulate nodes for this purpose. For this reason, our focus related to human–computer interaction was on providing menus and clicking options to achieve this goal. Moreover, the machine itself must be responsible for keeping links and their inverse consistent. From the way of doing knowledge acquisition we propose, documentation results as a "side effect". For keeping it consistent we utilise "active text". If the link structure is changed this way, in effect *dynamic* hypertext arises. All this is also useful for end-user support.

In general, we argued that knowledge-based systems can benefit. This is especially true when the knowledge bases become large. Therefore, Cyc's (Lenat, Guha, Pittman, Pratt, & Shepherd, 1990) very large knowledge base would probably be easier to handle and read if the original (natural language) text of the encyclopedia and the notes on the additional knowledge incorporated would be within the knowledge base, and linked to the operational objects.

Acknowledgments—The authors would like to thank Prof. Jörgl for his support of this project and for providing his domain expertise.

Thanks are also due to Holger G. Ziegeler for his aid in mastering TEX for the production of this paper.

REFERENCES

Bigelow, J. (1988). Hypertext and CASE. *IEEE Software*, **5** (2), 23–27.

Carando, P. (1989). Shadow: Fusing hypertext with AI. *IEEE Expert*, **4**(4), 65–78.

Carlson, D.A., & Ram, S. (1990). HyperIntelligence: The next frontier. *Communications of the ACM*, **33**(3), 311–321.

Conklin, J., & Begeman, M.L. (1988). gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, **6**(4), 303–331.

Fikes, R. (1988). Integrating hypertext and frame-based domain models. In *Proceedings of the AAAI-88 Workshop on AI and Hypertext: Issues and Directions*. AAAI.

Gaines, B.R., Rappaport, A.T., & Shaw, M.L. (1989). A heterogeneous knowledge support system. In B.R. Gaines & J.H. Boose, (Eds.), *Proceedings of KAW-89*, (p. 13/1–13/20). Banff, Canada. Menlo Park, CA: AAAI.

Gaines, B.R., & Sharp, M. (1987). A knowledge acquisition extension to NoteCards. In *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*. Reading, MA.

Garg, P.K., & Scacchi, W. (1989). ISHYS: Designing an intelligent software hypertext system. *IEEE Expert*, **4**(3), 52–63.

Gaschnig, J. *Performance measurement and analysis of certain search algorithms*. Unpublished Ph.D. thesis, Carnegie-Mellon University, Pittsburgh, PA. Available as Tech. Rept. CMU-CS-79-124.

Gruber, T.R. (1989). Automated knowledge acquisition for strategic knowledge. *Machine Learning*, **4**, 293–336.

Hayes, P.J. (1979). The logic of frames. In D. Metzing (Ed.), *Frame conceptions and text understanding* (pp. 46–61). Berlin: de Gruyter.

Hofmann, M., Schreiweis, U., & Langendörfer, H. (1990). An integrated approach of knowledge acquisition by the hypertext system CONCORDE. In *Proceedings of the European Conference on Hypertext (ECHT-90)-Hypertext: Concepts, Systems and Applications* (pp. 166–179), Paris. Cambridge: University Press.

Jordan, D.S., Russell, D.M., Jensen, A.-M.S., & Rogers, R.A. (1989). Facilitating the development of representations in hypertext with IDE. In *Proceedings of Hypertext* '89 (pp. 93–104), Pittsburgh, PA. New York: ACM.

Kaindl, H., & Ziegeler, H.G. (1992). HIS—An information system about hypertext on hypertext. *ACM SIGLINK Newsletter*.

Koh, T.-T., Loo, P.L., & Chua, T.-S. (1990). On the design of a frame-based hypermedia system. In R. McAleese & C. Green (Eds.), *Hypertext: State of the art* (Chap. 17, pp. 154–165). Oxford, England: Intellect.

Landow, G.P. (1989). The rhetoric of hypertext: Some rules for authors. *Journal of Computing in Higher Education*, **1**(1), 39–64.

Lenat, D.B., Guha, R.V., Pittman, K., Pratt, D., & Shepherd, M. (1990). Cyc: Toward programs with common sense. *Communications of the ACM*, **33**(8), 30–49.

Linster, M., & Gaines, B. (1990). *Supporting acquisition and interpretation of knowledge in a hypermedia environment*. Arbeitspapiere der GMD 455, Subreihe Künstliche Intelligenz 6, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Germany.

Ljung, L. (1986). *System identification theory for the user*. Englewood Cliffs, NJ: Prentice Hall.

Malone, T.W., Grant, K.R., Turbak, F.A., Brobst, S.A., & Cohen, M.D. (1987). Intelligent information-sharing systems. *Communications of the ACM*, **30**(5), 390–402.

Minsky, M. (1975). A framework for representing knowledge. In P. Winston (Ed.), *The psychology of computer vision* (pp. 211–277). New York: McGraw-Hill.

Nielsen, J. (1990). The art of navigating through hypertext. *Communications of the ACM*, **33**(3), 296–310.

Rochowiak, D. (1990). A tool for the management of document driven knowledge acquisition. In *Proceedings of the AAAI-90 Workshop on Knowledge Acquisition: Practical Tools and Techniques*, Boston, MA. Menlo Park, CA: AAAI.

Rochowiak, D., Ragsdell, B., & Wurzelbacher, L. (1989). An integrated hypertext and rule based system for explanation. In *Proceedings of the Expert Systems* '89 (pp. 345–352).

Walker, J.H. (1987). Document examiner: Delivery interface for hypertext documents. In *Proceedings of Hypertext* '87 (pp. 307–323), Chapel Hill, NC. New York: ACM.

Wells, T.L. (1989). Hypertext as a means for knowledge acquisition. *SIGART Newsletter*, **108**, 136–138.

Ziegeler, H.G., & Kaindl, H. (1991). A cyclic pattern resulting from a constraint satisfaction search. In *Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications (CAIA-91)* (pp. 373–344), Miami Beach, FL. Los Alamitos, CA: IEEE.