ELECTRONIC PUBLISHING, VOL. 3(4), 179–205 (NOVEMBER 1990)

A functional meta-structure for hypertext models and systems[1]

RICHARD FURUTA[2] AND P. DAVID STOTTS[3]

Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742–3255[4]

and

Office Systems Engineering Group
Systems and Software Technology Division
National Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899[5]

SUMMARY

We describe a hypertext 'meta-structure'—one that provides an organization for the architecture of hypertext models and systems. The meta-structure was initially developed to help us understand the architecture of a specific hypertext model (the Trellis hypertext model). However, its organization seems generally applicable to a wide range of other models and systems as well. As such, the meta-structure is a good candidate for a high-level hypertext *reference model***, and so we refer to it as the** *Trellis hypertext reference model***, or the** *r-model***. The r-model represents a hypertext at five levels of abstraction—two abstract levels, two concrete levels, and one visible level. In this paper, we present the r-model, use it to classify four different hypertext (and hypertext-like) systems, and then discuss its relationship to various hypertext-defined concepts.**

KEY WORDS Hypermedia Hypertext Meta-model Model Reference model Structure Trellis

1 INTRODUCTION

As a side-product of our work developing the Trellis model of hypertext [2] ,we have defined a 'meta-model' that provides an organization for the architecture of hypertext models and other highly interactive man/machine systems. It is the purpose of this paper to describe the meta-model within the context of Trellis and further to show how it classifies other hypertext models and systems. As such, the meta-model may serve as an appropriate framework for the development of a general hypertext reference model. As a reflection of this application, we refer to the 'meta-model' throughout this discussion as the *Trellis hypertext reference model*, abbreviated as *r-model*, A specific system or abstraction of hypertext itself will be called a *hypertext model*, or more simply a *model*, throughout the paper.

[1] This paper is an expansion of an earlier workshop presentation [1].
[2] Electronic mail: furuta@cs.umd.edu.
[3] Electronic mail: pds@cs.umd.edu
[4] The portion of this work carried out at the University of Maryland is based upon work supported by the National Science Foundation under Grant No. CCR–8810312. It is also supported in part by a grant from the Northrop Research and Technology Center. Please address correspondence to the University of Maryland address.
[5] Contribution of the National Institute of Standards and Technology. Not subject to copyright. The mention of commercial products in this paper is to describe the research environment and does not imply recommendation or endorsement by the National Institute of Standards and Technology.

0894–3982/90/040179–27$13.50
Received 16 August 1990
Revised 15 October 1990

The Trellis hypertext reference model is based around a collection of representations of the hypertext at different levels of abstraction. Abstractions range from the hypertext as a collection of abstractly defined independent components through more concrete representations in which the characteristics of the hypertext's physical display have been established, to the view of the hypertext that is projected on a physical display device for the benefit of the person reading the hypertext. The representations at a particular level of abstraction depend upon representations at a greater level of abstraction, and these dependencies are shown within the r-model.

A description of the r-model follows in the next section. Section 3 shows application of the r-model in categorizing the characteristics of a variety of hypertext systems. Next, Section 4 compares selected components of the r-model with components of some of the other models of hypertext, and discusses how the characteristics of those other existing hypertext systems and models fit into (or are omitted from) the r-model. Finally, we believe that the r-model can be used as a guide for developing new hypertext architectures in addition to a means for describing existing ones. Our experience suggests that architectures that closely mirror the r-model have some potentially desirable characteristics, such as the ability to easily extend the architecture to handle multiple interacting readers and writers. In Section 5, we conclude the paper by discussing some of the general characteristics and implications of the r-model.

2 THE R-MODEL

The r-model, abstractly diagrammed in Figure 1, is separated into five distinct levels. Within each level is found one or more representations of part or all of the hypertext. Speaking quite broadly, the levels may be grouped into three overall categories: abstract, concrete, and visible.[6] The abstract component and abstract hypertext levels define an abstract representation of the pieces of the hypertext and of the hypertext itself. These abstractions are transformed into more concrete representations of the hypertext in the concrete context and concrete hypertext levels, representing first the presentation of the hypertext's content and structure, and then the mapping of that information into displayed units. The resulting concrete windows are then viewed, each producing one *or more* displays on one or more physical devices. In summary, the representations in the abstract component level are at the greatest level of abstraction and those in the visible hypertext level are at the lowest level.

Each representation is shown in the figure as a box. A representation is itself an abstract concept—a consistent presentation of the hypertext elements of interest. Representations in the r-model may depend on the representations at a greater level of abstraction. Such a dependency is shown in the figure as an arc between the representations. Because a representation's dependencies are on those representations at a greater level of abstraction, and not on those at the same or lower levels of abstraction, the abstract and concrete levels in the diagram are further subdivided. It is worth emphasizing that a representation may not actually correspond to a separately identifiable 'physical' representation of the hypertext;

[6] The choice of these levels of representation parallels and expands Shaw's model of printed documents [3] which identifies abstract, concrete, and viewing mappings for the document.

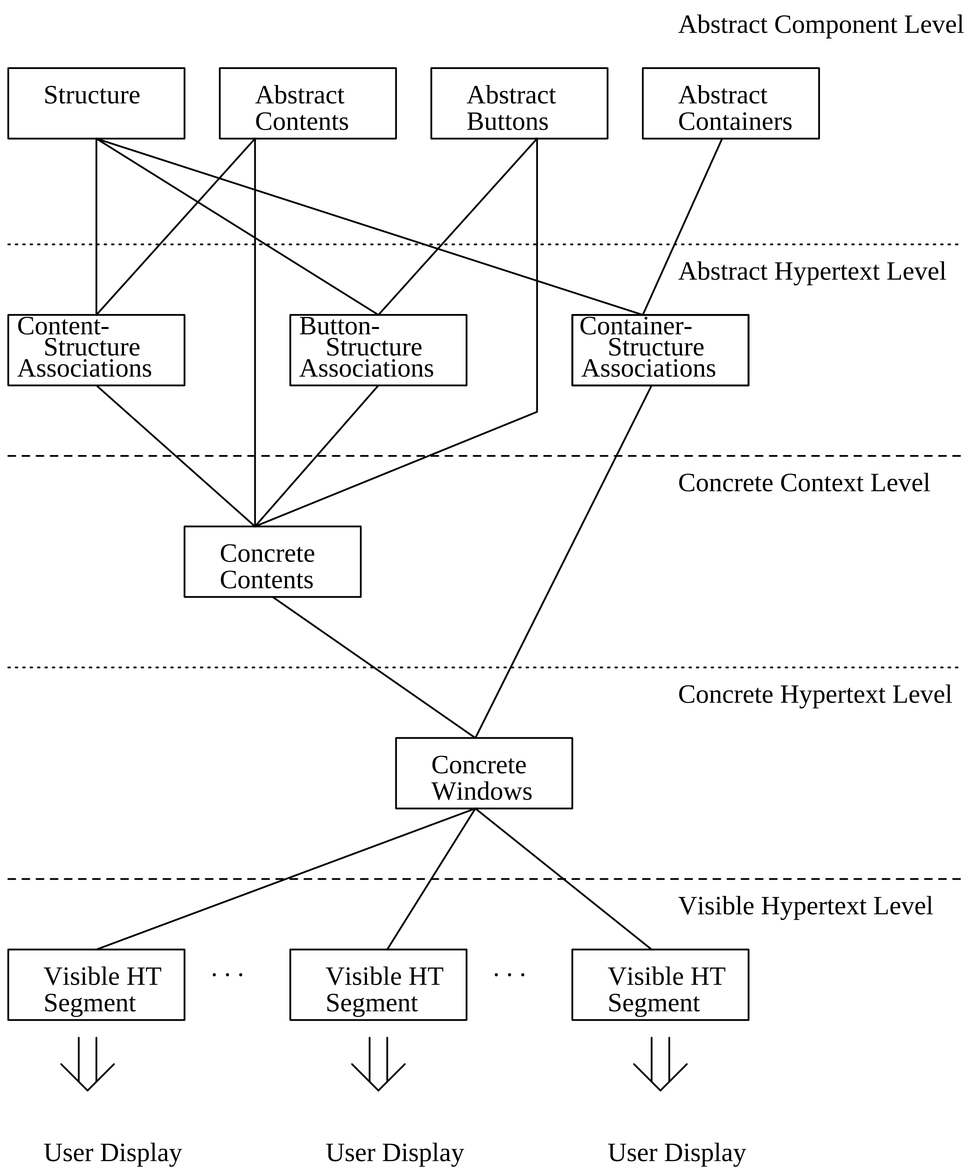


Figure 1. The Trellis hypertext reference model (the r-model)

for example, the representation may be expressed as a mapping between elements of more abstract representations.

We will now focus in turn on each of the levels of the r-model. In the following sections, we will describe the level and its representations, and discuss the dependencies on representations at higher levels.

2.1 Abstract layers

An abstract hypertext description specifies the logical structure and form of a hypertext and its components, but does not describe the particulars of how that hypertext is to be presented to its readers.

2.1.1 Abstract component level

The organization of the three highest levels reflects a separation of the hypertext into *structure*, *content*, and *context*. The structure represents the elements of the hypertext and their relationships. The specific content of the hypertext as presented to the system's user reflects the context within the structure in which the content appears—in other words, the display of the content is modified to reflect its context.

The representations within the abstract component level present the components that will be associated with one another to form the hypertext. Within the context of this level, the representations are independent of each other—such associations will be made at lower levels of abstraction. Our abstract view of a hypertext separates out the hypertext's structure from the elements that many users perceive as composing the hypertext. In other words, the structure (often a directed graph) is separated from the collection of contents that are to be displayed to the reader and the collection of 'buttons' that will be selected by the reader when moving from location to location in the hypertext. Additionally, it may be the case that the view of the hypertext presented to the reader combines together independent content elements into an integrated whole. The presence (or absence) of such composition is also represented abstractly at this level. We will now consider each of the representations in turn.

The most common, and arguably the most natural, representation for the **structure** of a hypertext is some form of directed graph, or network, either explicitly or implicitly represented.[7] In our Trellis model work, we use a Petri net for structure, which provides concurrent automaton semantics within a network representation. However, other graph-based structures are appropriate as well—for example, other automata such as deterministic finite automata and counter automata, or data structures such as trees, lattices, or semantic nets. The structure of the hypertext need not be limited to networks, though. Indeed, it may be desirable to use representations that are not ostensibly graph-based in form—constraint-based descriptions, for instance. Note that even in graph-based representations, there is no requirement that the elements of a structure be fully connected. The necessary

[7] A graph, however, is not the only possibility for a structural description. Graphs are necessarily finite; by using something like mathematical formulae, an infinite number of nodes and links can be expressed in a hypertext structure. See Section 3.4 for an example.

characteristics of a structural representation are that it provides the ‘placeholders’ that will be associated with the hypertext’s content elements, and that it describes any relationships that exist among these placeholders.

Abstract content. The abstract content is arbitrary in form. It may, for example, include textual, graphical, animated, or perhaps even audio and video material. The content may be specified directly or may be the result of a computation. While it does not contain links, it may incorporate markers that define a collection of potential locations for the mappings of links and their presentations that occur in lower levels of the r-model. The content may be described in a form that is independent of the eventual characteristics of its display, or indeed it may be described in a form that is highly dependent on the eventual display. Because of the flexibility of the mapping from content to structure in the next level, however, a display-independent representation seems most appropriate.

Abstract buttons. The structure representation identifies the relationships among content elements but does not indicate how those relationships will be shown for selection by the hypertext’s reader. The abstract buttons are representations of the ways in which the relationship can be displayed. Abstract buttons may themselves have content and an associated type. The content is provided to specify *what* will be shown when the button is displayed. The type is needed to specify *how* the button will be displayed and other characteristics of its behavior on display and selection. As with the content of the abstract content, the content of the abstract button is variable in form—in implementation it actually may be computed or it may be statically defined.

Abstract containers. The final component in this level, the abstract containers, differs from the others in that it is an abstraction of how the pieces of the hypertext will be combined when shown to the reader (*how* it will be aggregated and combined for display), and not of *what* is in the hypertext. Alternately, abstract containers can be thought of as extra-structural relations among elements in a document, that is, a type of ‘link’ that is not part of the browsable structure. For example, if several content elements are displayable, one possible presentation would be to show each element separately while another would be to combine the separate elements into a composite, which would be presented to the reader as a unit. In the first case, one could say that a separate container had been associated with each separate content element, while in the second case, one container would hold all content elements. This form of aggregation can be applied not only to content elements, but also to buttons and structure in general, as the next section explains.

2.1.2 Abstract hypertext level

The elements of the abstract component level are not connected together, as will be necessary to form a hypertext. This association is performed in the abstract hypertext level. The abstract hypertext level does not, however, describe *how* these associations will be presented within the display of the hypertext. This is left to the concrete context level.

Content–structure associations. The content–structure associations map together elements of the structure and elements of the abstract content. In a graph-based structure, one natural association is to map the content elements to the nodes of the graph. No restriction is expressed in the r-model on the kinds of mappings that are permissible—for example it may be useful to map a single content element to multiple locations in the structure, or conversely to map multiple content elements to a single location. In our own work, we have found the ability to map a single content element to multiple locations to be particularly useful. We have also found it useful to completely substitute a new collection of abstract contents and of content–structure associations while retaining the same structure—for example for related hypertext versions, where one may perhaps be a translation of the other.

Button–structure associations. The button–structure associations map the structure's relationship and abstract buttons. A natural association in a graph-based structure is to map the abstract buttons to arcs in the graph. In our Trellis hypertext model, based on Petri nets, the mapping is between the class of node called a transition and the abstract buttons (i.e., there is no mapping of arcs in this particular graph structure). Again we emphasize that there are no limitations expressed on the form of the mapping, although we have found a one-to-one mapping to be the most useful.

Container–structure associations. Finally, the container–structure associations describe the association of the structure, or of portions of the structure, to one or more abstract containers. One use of this association is to permit grouping of elements of the structure, which might in turn be displayed to the reader in a single physical window. Different kinds of composite displays would be represented as associations with different types of abstract containers. In general, the container–structure associations allow the partitioning of the subsequent display of the hypertext into one or more possibly overlapping pieces.

2.2 Concrete layers

Assume that a hypertext is presented to its reader or readers in one or more windows on one or more physical display devices.[8] A concrete hypertext description specifies what the contents of each of these windows will look like but does not tie down how the windows are to be arranged on the display. For example, one particular window may be shown on several separate displays. Furthermore, the characteristics of the displays may be different; in this case the subsequent viewing description will also indicate how the different visible effects specified by the concrete description are to be rendered on the displays.

2.2.1 Concrete context level

The previously described levels have defined an abstract hypertext in which the content and the buttons have been associated with the structure. However, the abstract hypertext description does not indicate how links are to be presented in the display of the content. Such considerations of the mapping from the hypertext's abstract representation to its physical representation are addressed in the concrete context level.

The concrete content presents a physically oriented description of the hypertext. This mapping must address the following points:

- How is the abstract content to be formatted to fit within the display region?

[8] Here a window contains a concrete view of the hypertext (or portion of the hypertext) to be presented to the reader.

- How are the buttons to be displayed? Will the display of the button modify the display of the content or will the buttons and content be displayed independently? For example, in our initial Trellis prototype (αTrellis), we have provided externally represented buttons. In our subsequent prototype (χTrellis), we have also developed means for specifying that the button is to be represented as a highlighted string within textual context [4]. Note that button displays are not necessarily static; in some cases the display of the button depends on computed material (which itself may depend on the structural relationships in the hypertext). The button represents the source of a link in the hypertext.[9]
- Is the target of a link associated with a content element as a whole, or is it associated with a particular location within that content? Does the display of the target affect the display of the content?

The mappings on this level do not rely directly on the structure (abstract component level) because the structural relationships have been 'encoded' into the representations of the abstract hypertext level.

2.2.2 Concrete hypertext level

The concrete context level has defined a set of concrete content elements in which a concrete representation of the content has been merged with concrete representations of the buttons. The concrete hypertext level maps those concrete representations into a set of windows for display. The mapping, which produces the **concrete windows** representation, also requires that link-based interrelationships among the windows be determined. For example, the process of following a link can result in several different display mappings: the display of the target of the link could replace the display of the source, could be shown in addition to the source, or could modify the display of the source, with both being shown in the same window.

When the concrete windows representation has been formed, the presentation of the hypertext has been determined but the details of how and where the windows are to be displayed has not. For example, multiple windows may be shown to a single reader on a display or a particular window may be shown to several readers simultaneously on separate displays. Indeed, a particular reader may have several physical displays at his disposal, and different displays may have equivalent but different means for achieving particular visual effects. Such considerations are addressed in the next level.

2.3 Visible layers

The details of the mapping from the concrete hypertext to the consumable presentation of a hypertext for readers are specified here.[10] However, user interface details, such as the positioning and sizing of windows, are orthogonal to the r-model, as discussed later in this paper.

[9] See also the comparison with anchors that follows in Section 4.1.2.

[10] Although we use words like 'visible' we do not intend to imply that the r-model applies only to text, graphics, or other visual media. Presentation of other forms of media, such as audible, physical, etc., and management of forms of computation not visible to the reader, would follow along the same lines.

2.3.1 Visible hypertext level

An assumption in the r-model is that the underlying hypertext is to be permitted to be used in a distributed environment. The visible hypertext level reflects this assumption. Each **visible HT segment**[11] may be associated with a separate user and/or display (or other physical device). Each segment presents one or more of the active concrete windows to its viewer. The model does not prevent the display of a particular concrete window in more than one segment. Whether (and how) the effects of user interactions to one display may affect what is shown on other user displays is a property of the hypertext model, and not of the r-model.

3 EXAMPLE SYSTEMS AND CLASSIFICATIONS

In this section, we will use the r-model to classify the characteristics of a number of hypertext systems. We will begin by examining the similarities and differences between the Trellis Hypertext model and the r-model. We will next examine two popular PC-based hypertext implementations. The first, Hyperties for the IBM-PC, is a simple but comprehensive system that lends itself to a fairly direct classification. The second, HyperCard for the Apple Macintosh, is a more complex system with strong characteristics of a procedurally based visual program creation environment. We believe that examination of HyperCard and comparison of its model with the r-model will help to better demonstrate our motivations in designing the r-model. Finally, we examine the r-model classification of some image browsers (e.g., fractal browsers). Although these systems may not appear at first glance to be hypertextual systems, we believe that they share features in common with systems that are commonly believed to be hypertextual. Consequently examination of their r-model classification provides both a basis on which an extended notion of ‘hypertext’ can be defined, but also provides a framework on which implementation of such extended hypertexts can be designed.

3.1 Trellis

Since the original characteristics of the r-model were developed by considering the similarities and differences between our Trellis hypertext model [2,5] and other hypertext models, it is instructive to first consider the r-model categorization of the Trellis model (summarized in Figure 2). The Trellis model is based on the formalism provided by a timed Petri net. The prototype hypertext browsing and authoring system αTrellis is based on the model.

An important characteristic of the Trellis model and also of the αTrellis prototype is that the relationships among objects closely parallels those in the abstract levels of the r-model. As a result, hypertextual content elements are defined separately from the hypertext’s structure. The same content element can appear in different locations within the hypertext. Issues involving the appearance of these different elements are addressed by the Trellis model’s logical to physical projection, P_d, which is categorized in the r-model’s concrete levels. An additional consequence of the strong separation of content and structure is that

[11] Here, ‘HT’ is an abbreviation for ‘hypertext.’

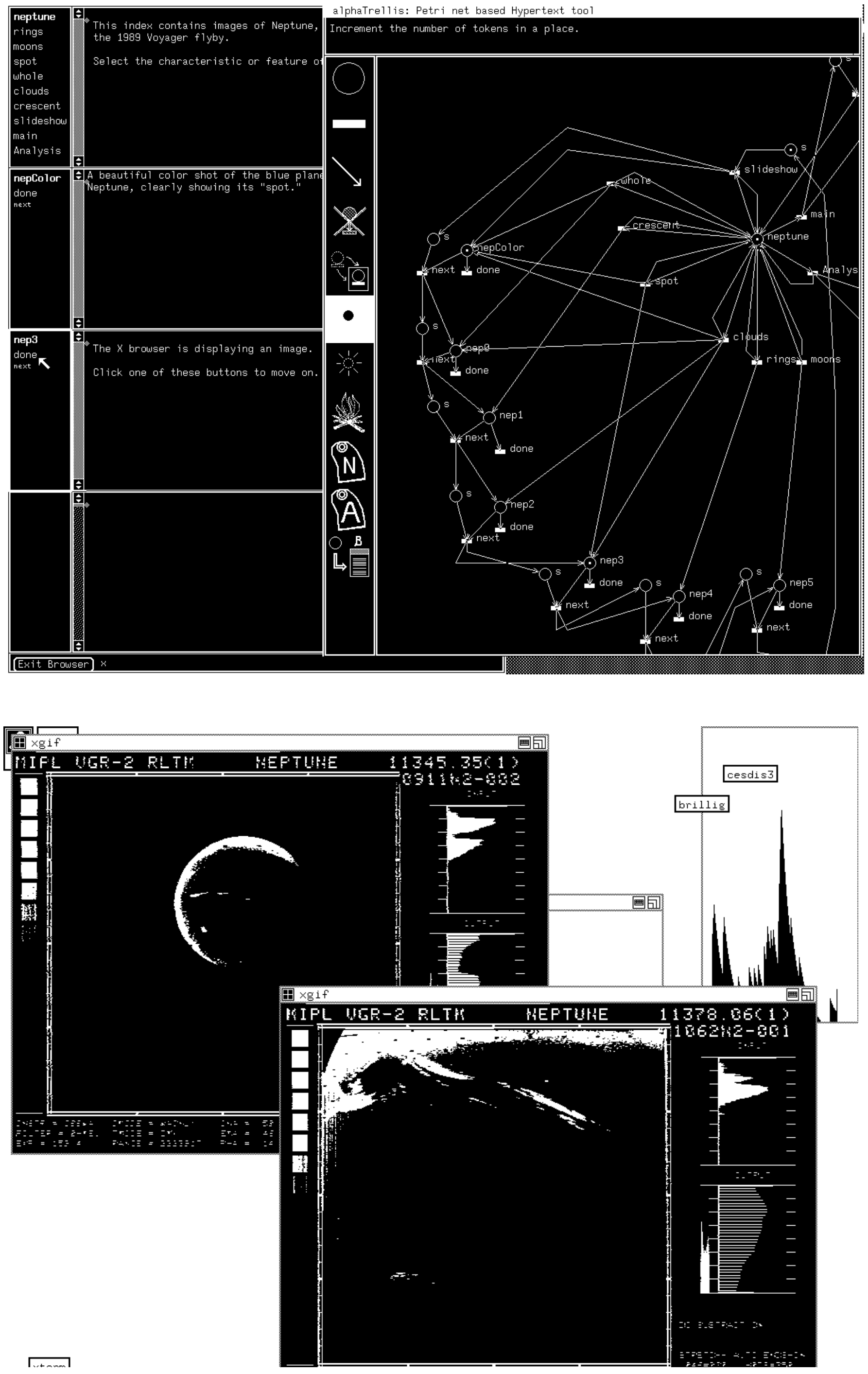
level	*function*
abstract components	STRUCTURE: Described by a timed Petri net, N, and an initial marking M_0. CONTENTS: Not limited by the model. Can also include other hypertexts. Described as a set, C. BUTTONS: Not limited by the model. Described as a set, B. CONTAINERS: Can contain (encapsulate) one or more sets of contents and one or more collections of buttons. Described as a set, W
abstract hypertext	The functions on this level are described by the logical projection P_l. CONTENT–STRUCTURE: Content element associated with each place in the Petri net (mapping C_l in the Trellis model). (Note that a given content element can be associated with many or no places.) BUTTON–STRUCTURE: Button or buttons associated with each transition in Petri net. However, each transition can have at most one button associated with it. In prototype implementation, the mapping is one-to-one (mapping B_l). CONTAINER–STRUCTURE: Maps an abstract container to each place in the Petri net (mapping W_l).
concrete context	Represented directly in Trellis prototypes. Specified as a portion of the P_d projection in the Trellis model. In αTrellis, buttons and content are displayed independently.
concrete hypertext	Specified as a portion of the P_d projection in the Trellis model.
visible hypertext	Implemented as multiple clients in Trellis prototypes. A client corresponds to a visible hypertext segment. Each concrete window is mapped to zero or more visible hypertext segments. Each visible hypertext segment displays zero or more concrete windows. Zero or more clients may be mapped to a particular hardware device. Each client may control zero or more devices (generally one).

The definitions of the Trellis model components are taken from Reference [2].

Figure 2. R-model classification of **Trellis**

different collections of content elements can share the same hypertext structural definitions (e.g., different ‘versions’ of a hypertext presented perhaps in different languages).

Figure 3 ,shows a snapshot of the αTrellis prototype in action. αTrellis’s implementation is based upon a client-server model. The server implements the ‘Trellis engine’—i.e., the semantics of the Trellis model as based on the timed Petri net specification. The clients



This index contains images of Neptune, the 1989 Voyager flyby.

Select the characteristic or feature o

A beautiful color shot of the blue plane Neptune, clearly showing its "spot."

The X browser is displaying an image.

Click one of these buttons to move on.

alphaTrellis: Petri net based Hypertext tool

Increment the number of tokens in a place.

Figure 3. αTrellis display on two separate screens

implement interfaces to humans and to external computer programs. Exactly one server must be running, but any number of clients (zero or more) may be active.

The snapshot in Figure 3 shows three clients distributed across two different displays. The topmost display contains a Petri net editor on the right and a text browser display on the left. The Petri net editor client is permitted both to display and also edit the structure of the hypertext. The text browser client can only display the hypertext. The lower display shows images displayed on a color screen by a graphics display client.

Each client runs in its own environment and makes its own determination of what portion of the hypertext's active content elements it will display and how it will represent any links associated with the elements. For example, the clients in the upper display are running under the SunTools window system and the client in the lower display is running under X windows, release X11R3. The text browser client in the upper display is showing all nodes with textual content plus placeholders for nodes with graphical content and is showing links in a button panel to the left of the textual display. The graphical display client in the lower display is showing only those nodes with graphical content and is not showing any buttons.

Note that separate clients may be instances of the same class—for example if two text browser clients were running and displaying their output on the upper display, two copies of the text display would be visible. This also permits replication of a particular display onto separate (distributed) workstations. Note also that the clients' ability to selectively choose the types of content elements to display provides an easy way to represent and manage multimedia presentations requiring independent output devices.

The Trellis hypertext model and also the r-model classifications tend to favor thinking of the hypertext's architecture as described by a collection of abstract data types. This leads to an implementation that makes the distributed client-server applications easy to provide. The Trellis model follows the r-model distinctions fairly closely, as shown in Figure 2. However, Trellis's physical display projection (P_d), which describes how the abstract objects of the hypertext are to be formatted and combined to produce a concrete representation, is further refined into two levels in the r-model.

3.2 Hyperties

In this section, we will consider the classification of a relatively simple, but comprehensive hypertext system, Hyperties [6], which runs on the IBM-PC. The discussion of Hyperties is based on version 2.35.[12] The appearance and characteristics of the to-be-released version 3.0 differ. The r-model classification of Hyperties is summarized in Figure 4. A hypertext in Hyperties is said to be a *database* of *articles*. Only one article is visible to the reader at a time.

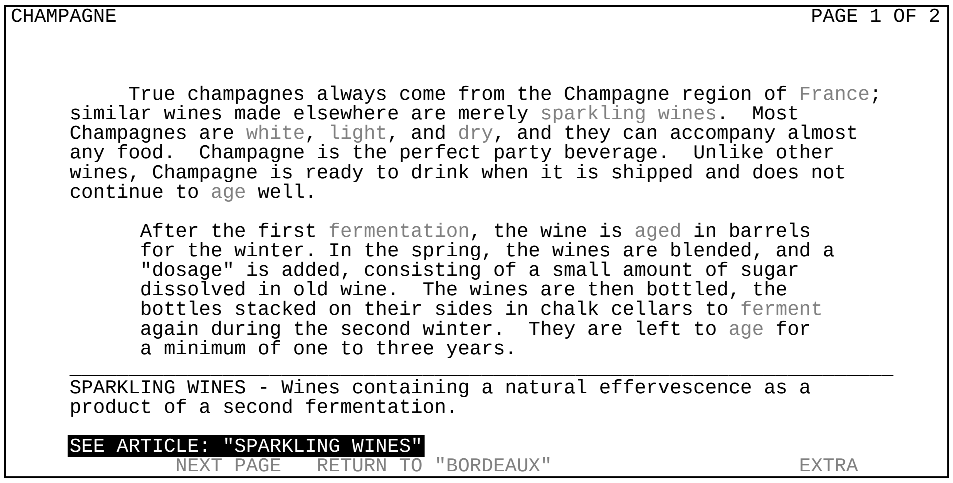
Figure 5 shows a screen displaying an article from a Hyperties database.[13] The half-toned words and phrases represent buttons. Article-specific buttons are shown above the line in the displayed screen. Selecting a button will bring up a short summary of contents of the target article, shown just under the line in the figure, and will display a button that if

[12] A description of the internal data format used by this version of Hyperties can be found in Reference [7].

[13] This screen is taken from a Hyperties database titled 'Wines of France,' written by Elizabeth Buie and Ed Hassell.

layer	*function*
abstract components	STRUCTURE: Described by a directed graph. CONTENTS: Two kinds of content elements: textual and graphical. Elements are uniquely named (i.e., in implementation by their DOS file name). Textual elements, called articles, are in turn subdivided into article title, article body, and summary. BUTTONS: Two categories: article-specific (associated with a specific article) and system-supplied (always present when relevant). CONTAINERS: Three types of separate sub-containers. The sub-containers represent respectively holders for textual content, graphical background, and system-supplied buttons. The textual content sub-container is further subdivided into fields that eventually will hold the article's title, and content body. The system-supplied button sub-container holds both the text of those buttons and also the summary of the target of a link.
abstract hypertext	CONTENT–STRUCTURE: Structure is encoded in the representation of content elements, and is not represented separately. Hence, the content element is associated with a single node in the structure. BUTTON–STRUCTURE: Button associated with edge in graph. Link structure is encoded in the representation of the source content element as a reference to the target's unique identifier. CONTAINER–STRUCTURE: A container is associated with each node in the graph.
concrete context	Textual content is formatted by applying directives written in a specialized formatting language. Buttons are represented as highlighted strings; specification of these strings is part of the markup defining the formatted content. The target of the link is the node as a whole; there is no mechanism to specify that the target is located at a particular point in the node's content.
concrete hypertext	A single concrete window is defined. The system-defined sub-container contains buttons and fields that are always present, if appropriate in context, and is always displayed. Either textual material or graphical material may be omitted, but at least one must be present. If both are present, the text is displayed on top of the graphics.
visible hypertext	There is only a single visible HT segment, which displays the single defined concrete window.

Figure 4. R-model classification of **Hyperties**



```
CHAMPAGNE                                                          PAGE 1 OF 2


          True champagnes always come from the Champagne region of France;
     similar wines made elsewhere are merely sparkling wines.  Most
     Champagnes are white, light, and dry, and they can accompany almost
     any food.  Champagne is the perfect party beverage.  Unlike other
     wines, Champagne is ready to drink when it is shipped and does not
     continue to age well.

           After the first fermentation, the wine is aged in barrels
           for the winter. In the spring, the wines are blended, and a
           "dosage" is added, consisting of a small amount of sugar
           dissolved in old wine.  The wines are then bottled, the
           bottles stacked on their sides in chalk cellars to ferment
           again during the second winter.  They are left to age for
           a minimum of one to three years.
     ______________________________________________________________________
     SPARKLING WINES - Wines containing a natural effervescence as a
     product of a second fermentation.

     SEE ARTICLE: "SPARKLING WINES"
              NEXT PAGE   RETURN TO "BORDEAUX"                     EXTRA
```

Figure 5. A sample Hyperties display

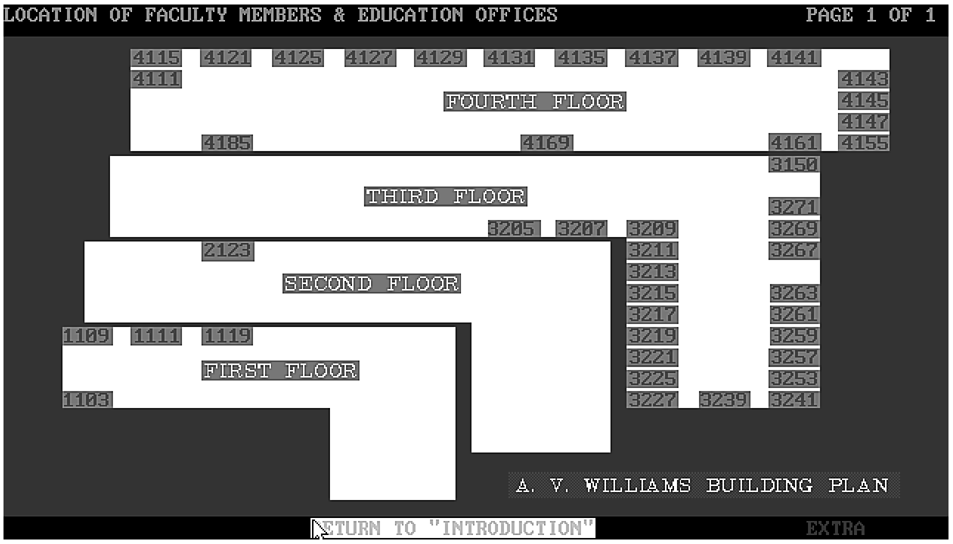


Figure 6. A Hyperties display using a graphical background

selected will replace the current display with that of the target display. This button is the default that will be selected in the absence of further positioning commands, and so it is shown in reverse video here.

In addition to textual content, a displayed screen may also contain graphical content. If both textual content and graphical content are specified, the graphical pattern serves as background for the textual material. This can serve the purpose of providing some sort of appropriate background for textual material. Additionally, as in the case illustrated in

Figure 6,[14] text from the textual content can be arranged so that it overlays the graphical information in a meaningful way. As example, the room numbers in this figure are selectable and lead to articles describing the rooms' occupants.

Although the generally available version of Hyperties is based on the IBM-PC, separate research versions are also being developed based on Sun workstations [8]. Some of the research versions have incorporated multiple windows, but for the most part, the multiple windows are independent of one another—i.e., browsing in one window does not affect the state of any other. Lifshitz [9] has described a series of experiments using a multiple-window version of Hyperties in which the display resulting from following a link can be shown in a different window from the source. Invocation of a window's display therefore can be related to actions taken in a different window, but once invoked, actions taken in a window are independent of the actions taken in the others, with the proviso that the result of one window's action may obscure the display in another window.

From the perspective of the r-model, the effect of the multiple-window displays as used in these research prototypes is reflected in the concrete hypertext level and in the visible layer. Such replicated displays would be modeled by also replicating the concrete windows and mapping them to separate visible HT segments.

3.3 HyperCard

The r-model classification of HyperCard [10,11], which is provided for the Apple Macintosh, is summarized in Figure 7. The discussion in this paper is based on Version 1.2 of HyperCard. Two sample Hypercard screens are shown in Figure 8. These screens were created by Judi Moline [12,13], and are from an inventory of Arab coins dated about 690 A.D. The figure shows two sides of the same coin.

Figure 8 illustrates a number of the characteristics of HyperCard. These screens share a common background, containing, for example, the button labeled 'TOC' in the lower left-hand corner of the displays. The material specific to the surface of the coin being described is contained on a 'card' that is specific to the coin and side of the coin being presented. For example, the buttons above the 'TOC,' the drawing of the coin, etc., are located on the specific card.

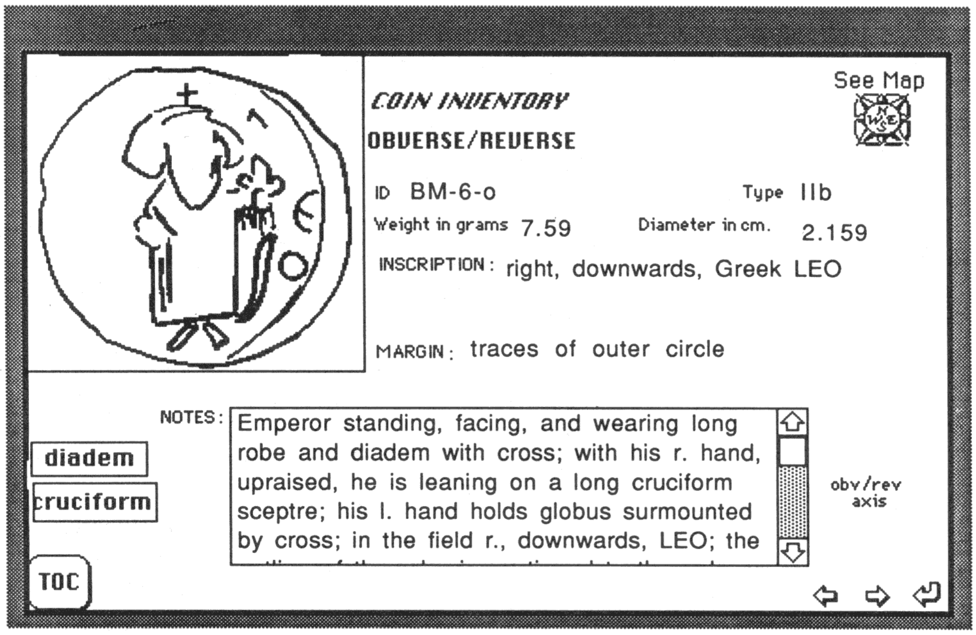
A HyperCard-based hypertext is formed from a collection of *cards* organized into *stacks*. One card is being displayed and is called the 'current card.' HyperCard defines a number of objects: buttons, fields, cards, backgrounds, and stacks, to be precise. A 'script' (in essence an executable procedure) may be associated with each object. Selecting, for example, a button on a card causes a message to be generated which is passed along a path until a script is found that defines a handler for that kind of message. A static path starts with the script at the button or field, then inspects the script at the card, at the background, at the stack, and at a distinguished stack called the home stack. If no handler is found, HyperCard itself handles the message.

A HyperCard script also can send a message to a card in a different stack or can cause the focus to transfer to another card, either in the same or in a different stack (cards can be addressed uniquely). This defines the dynamic path, which may cause the execution of handlers not found in the static path.

[14] The screen in Figure 6 is taken from a database presenting the University of Maryland's Computer Science Department. The database was written by Yoram Kochavy and Jasmin Naraine and subsequently updated by Chris Williamson.

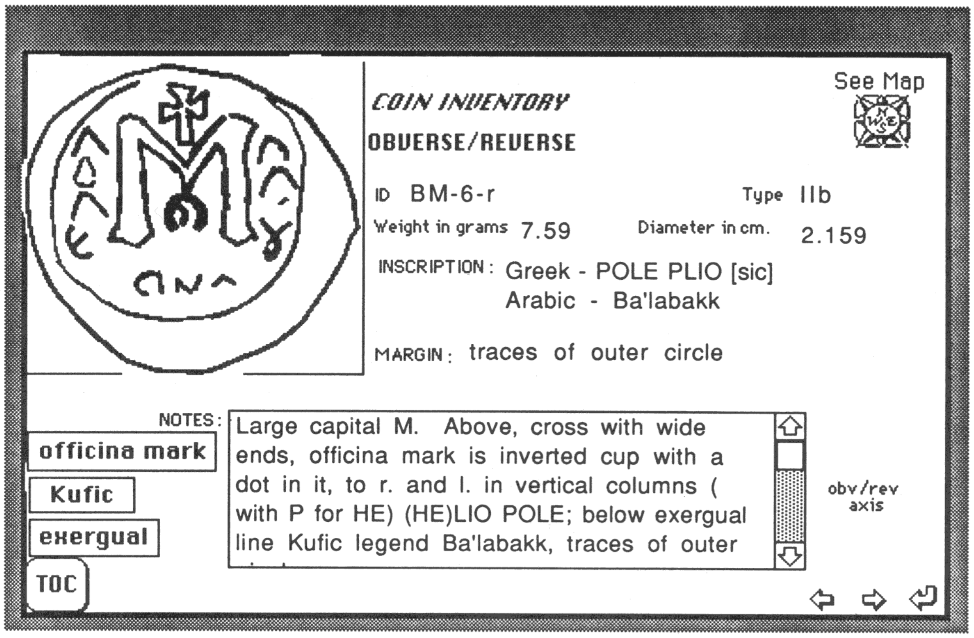
layer	*function*
abstract components	CONTENTS: pictures, buttons and fields. Pictures are bitmaps, buttons will become selectable and can contain either bitmaps or text, and fields will contain editable text. BUTTONS: The HyperCard button incorporates parts of both of the r-model abstractions of button and of content since it represents both a mechanism for effecting traversal along the structure but also information that is presented to the hypertext's reader (e.g., a selectable region of text within a paragraph is actually represented as a HyperCard button). CONTAINERS: There is a single type of abstract container that represents the display of a *card* and the display of a *background*. The card and background can be thought of as sub-containers within the container. (The r-model container is not the same as the HyperCard container.) STRUCTURE: Two structures are associated with HyperCard. An implicit structure, which we will call the link structure, represents link interrelationships among elements of the hypertext, which is the result of carrying out the computations associated with those elements. An explicit structure, HyperCard's static and dynamic paths, represents how those interrelationships are to be effected. See the discussion for details.
abstract hypertext	CONTENT–STRUCTURE: The content is formed by placing content elements onto cards and onto the background. The link structure is implied by the association of card and background and by the specifications associated with those elements and by those along the static and dynamic paths. BUTTON–STRUCTURE: R-model buttons are associated with the HyperCard implicit link structure by a dynamic mapping. CONTAINER–STRUCTURE: A container is associated with a card and a background.
concrete context	Fields in the display are positioned and sized by the author. Content within the fields is then formatted, clipped, or placed within a scrolling area as necessary to fit the available space.
concrete hypertext	A single concrete window is defined. If both a card and a background are present, the card's material is displayed on top of the background's material.
visible hypertext	There is only a single visible HT segment, which displays the single defined concrete window.

Figure 7. R-model classification of **HyperCard**



COIN INVENTORY
OBVERSE/REVERSE

ID BM-6-o
Type IIb
Weight in grams 7.59
Diameter in cm. 2.159
INSCRIPTION: right, downwards, Greek LEO
MARGIN: traces of outer circle
NOTES: Emperor standing, facing, and wearing long robe and diadem with cross; with his r. hand, upraised, he is leaning on a long cruciform sceptre; his l. hand holds globus surmounted by cross; in the field r., downwards, LEO; the



COIN INVENTORY
OBVERSE/REVERSE

ID BM-6-r
Type IIb
Weight in grams 7.59
Diameter in cm. 2.159
INSCRIPTION: Greek - POLE PLIO [sic]
Arabic - Ba'labakk
MARGIN: traces of outer circle
NOTES: Large capital M. Above, cross with wide ends, officina mark is inverted cup with a dot in it, to r. and l. in vertical columns (with P for HE) (HE)LIO POLE; below exergual line Kufic legend Ba'labakk, traces of outer

Figure 8. Sample HyperCard screens

From the point of view of hypertext, the target of a link is a particular card and the source is associated with a button (or field) on the current card. The determination of what card is to be the target (i.e., the means by which link traversal is effected) is as the result of executing the code found along the path.

HyperCard's strongly object-based model incorporates some elements that are not too commonly found in other hypertext systems. For example, the identification of card, background, and stack as objects in the HyperCard model and the association of executable specifications that can alter the display's state with cards, backgrounds, and stacks, along with the path mechanisms, means that the *containers* play a much more central role in what is presented to the HyperCard reader than in the other systems being discussed. In other words, the actions associated with containers can affect the link structures perceived by the reader, and hence the container relationships must be identified in the abstract component layer's structure.

3.4 Fractal browsers

As a final example, we present the r-model classification of a common, but perhaps surprising, hypertextual form (summarized in Figure 9). Image browsers for the Mandelbrot set, Julia sets, and other fractals have become popular in recent years. These tools present

layer	*function*
abstract components	STRUCTURE: Real Cartesian plane; infinite number of nodes (and so an infinite number of links); very large arity at each node CONTENTS: Bitmap computed from mathematical formula BUTTONS: Infinite set of buttons CONTAINERS: Single container
abstract hypertext	CONTENT–STRUCTURE: Subplanes and resolvable (x, y) points determine parameters for image computation from formula BUTTON–STRUCTURE: one button for the current subplane (*zoom*), and two buttons for each resolvable (x, y) point (*center* and *new*) CONTAINER–STRUCTURE: entire structure mapped to the single container
concrete context	*zoom* mapped to right mouse button, anywhere in frame; *center* mapped to center mouse button, on the pixel for each (x,y) point; *new* mapped to left mouse button, on the pixel for each (x,y) point
concrete hypertext	one window created to clip and frame a current subplane from the whole; buttons have no concrete representation other than the image pixels
visible hypertext	single instance of window, mapped to one workstation only

Figure 9. R-model classification of the fractal browser **doily**

a visual representation of some mathematical space, and allow a browser to outline with the mouse a rectangular region for further explorations. Another image is then created and presented for the indicated subspace. This form of browser is not limited to fractal equations, but also occurs in the context of traditional charts, Cartesian coordinate plots, and other data representations. For example, consider the public domain software tool called *xgraph*. Figure 10 shows a screen from *xgraph* in which a data set has been plotted, and a subrange outlined and enlarged for further examination. In this tool, both the source and the target of the 'enlarge' link are visible at one time.

Though not normally thought of as hypertext, the direct-manipulation aspects of image browsers like this can be understood in hypertextual terms through the r-model. The abstract structure in this case is a mathematical formula rather than any form of directed graph. For discussion, let us consider this formula to be a fractal defined on the real Cartesian plane [14], consisting of the set of (x, y) pairs iteratively defined by these equations:

$$\begin{aligned} x_0 &= y_0 = 0 \\ x_{i+1} &= y_i - sign(x_i)\sqrt{|13x_i - 1400|} \\ y_{i+1} &= 5 - x_i \end{aligned}$$

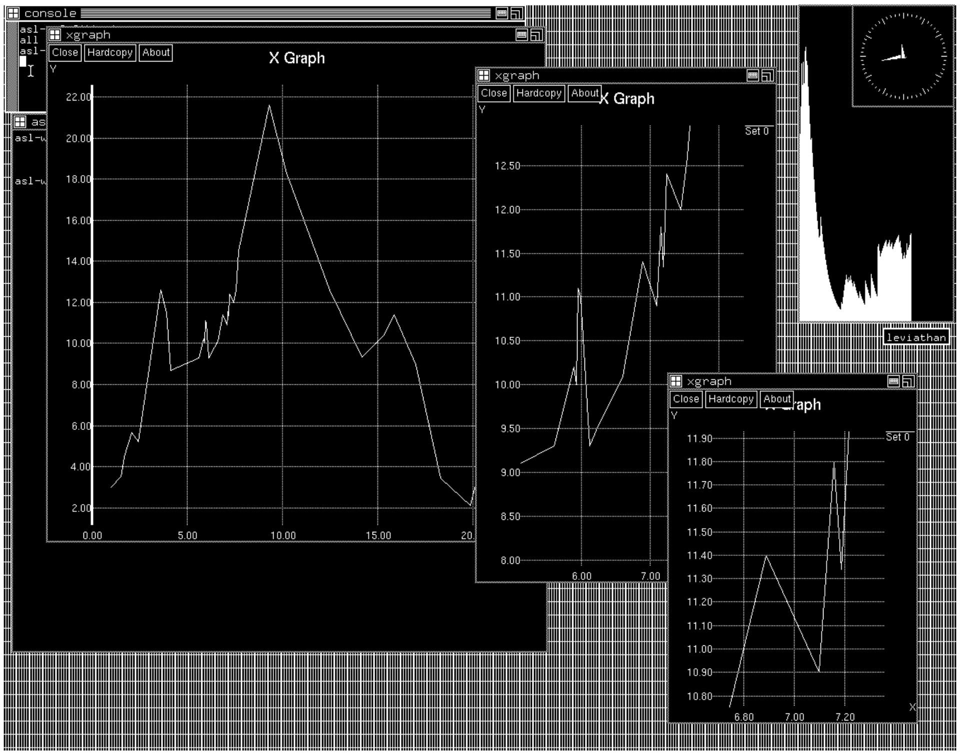


Figure 10. Screen from the xgraph formula browser

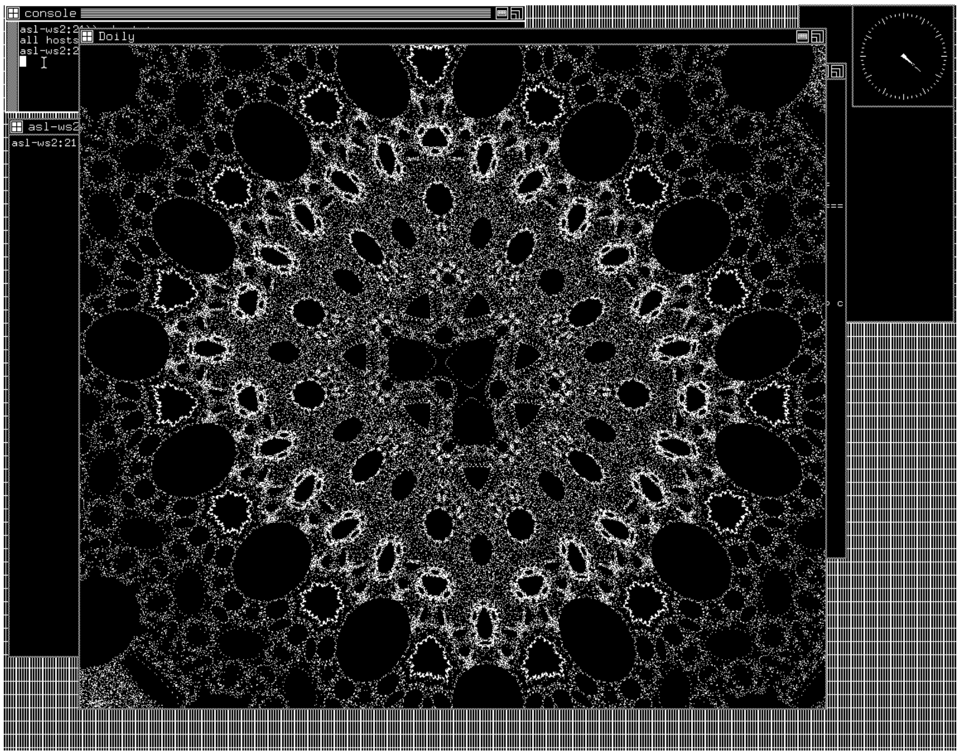
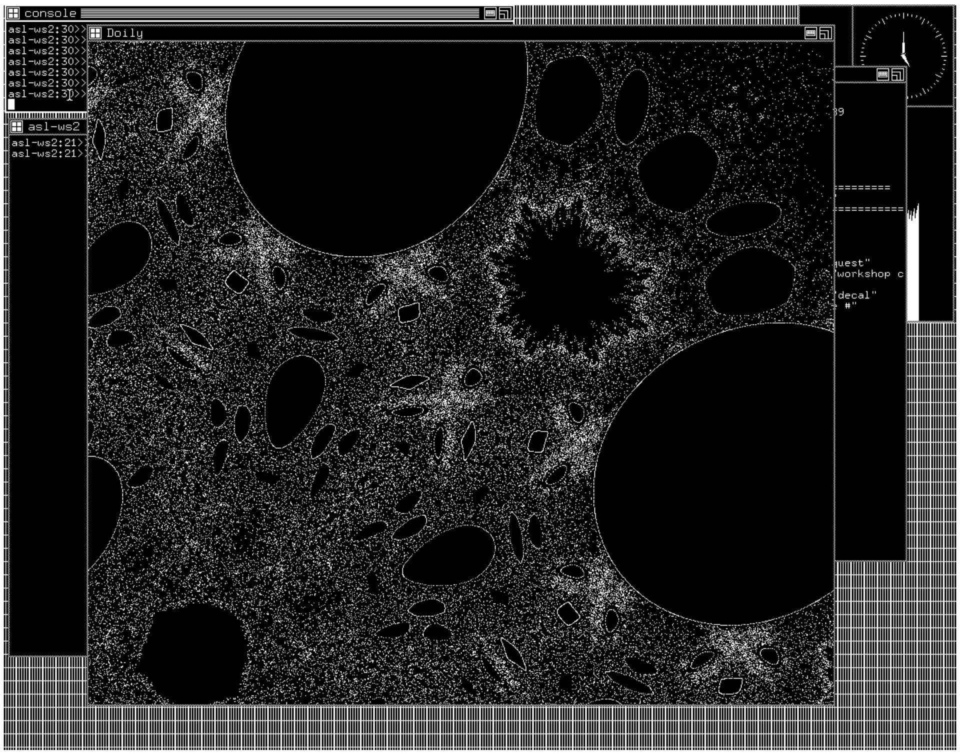


Figure 11. Screen from the doily fractal browser

Each content node in the browsable structure for this fractal is then an image computed from the equations using a particular set of bounding parameters for the field of view. A link is, in essence, any rectangular subregion within an image, and the target of a link is the new image computed from the fractal equation with the parameters for the link's defining rectangle. The fractal is defined on the real plane, which is dense, so there are an uncountably infinite number of nodes implicitly defined in this hypertext.[15] Each node has a very large, but nonetheless finite, number of links leading out of it (defined by the number of distinct rectangles within the image display window).

Figure 11 ,shows a browser we have implemented for this fractal, which we refer to as *doily*. In this screen, the image has been computed and accumulated with the initial point $(0, 0)$ in the center of the image, and a field of view limited to ± 10 in each direction. The second screen shown in Figure 12 shows another image produced from the same fractal, but with the center shifted to about $(5, 3)$ and the image magnified by a factor of 4 (that is, the field of view cut to around ± 2.5 units in either direction from the new center). In *doily,* this transition from the first image to the second occurs in response to clicking the middle mouse button on the point that is to be the new center, and then clicking the right mouse button twice to double the magnification twice. Thus, in r-model terms, every (x,y) point

[15] A Mandelbrot browser would also have an uncountably infinite number of nodes, since its fractal equation is defined on the complex plane. Equations defined on discrete spaces, like an integer Cartesian plane, would, in comparison, have give rise to a *countably* infinite number of nodes in their r-model structure.



```
asl-ws2:30>
asl-ws2:30>
asl-ws2:30>
asl-ws2:30>
asl-ws2:30>
asl-ws2:30>
asl-ws2:30>
```

```
asl-ws2:21>
asl-ws2:21>
```

Figure 12. Center shift and magnification with doily buttons

resolvable on the screen is a button (two in this case), and the entire frame is a button as well.

The context level is fairly simple in a fractal browser, since few buttons, if any, are explicitly represented. In *doily*, no explicit representation is made of any buttons other than the pixels themselves in the fractal image. In other browsers, perhaps a ‘return’ button might be needed to ‘unzoom’ back to a parent image. Such buttons can be mapped at the context level to a predefined menu area on the edge of the display frame.

The visible levels are also simple. A fractal browser may maintain one window and replace each image with the target of a selected link, as in *doily*. A browser may, alternately, create a new window for each target selected, maintaining the parent image concurrently, as in *xgraph*. In this second case, the ‘return’ button would be instead a ‘done’ button, since the entire traversal tree is explicitly maintained on the screen.

4 ISSUES IN APPLICATION OF THE R-MODEL

We now turn our attention to three specific aspects of the r-model, which we shall consider in detail. In Section 4.1, we discuss some important components of hypertext systems and how they fit into the r-model. In Section 4.2, we turn our attention to central issues in implementation of a hypertext system that are orthogonal to, that is, are not described by, the r-model. Finally, in Section 4.3, we discuss the intersection of our r-model with existing standards, both defined and de facto.

4.1 Aspects addressed by the r-model

A number of concepts, structures, and components have been identified for hypertexts.[16][12] Here, we present some of these hypertext elements and describe their categorization within our reference model.

4.1.1 Hypertext model structures

Many hypertext models concentrate on the benefits of some specific structure for linking information elements. As stated earlier, the abstract structure of a hypertext has its form neither prescribed nor proscribed within the r-model. Document structure may be graph-based, describing only object interrelationships, as is commonly used. It may additionally have automaton semantics, like the Petri net in Trellis [2], or the hypertext abstract machine (HAM) [16]. It may be intended to provided an inferential foundation, like a semantic net. It may be some generalization of a directed graph, like a hypergraph [17]. We have even seen previously how a mathematical formula can define the links in a hypertext.

The specific structure used in a model or system need not be homogeneous in form; heterogeneous structures may be appropriate for some applications. Structure need not be static in form but may be dynamic. Indeed, it need not be explicitly computed or represented. What is required, however, of an abstract structure is that it be possible to intuit where content can be included in a hypertext, as well as to determine the relationships among content elements.

4.1.2 Anchors

Anchor is the commonly used term for the terminal points of a link. It additionally conveys the sense that such a terminal point identifies an ‘interior’ portion of a content element—a word or phrase, a feature in a diagram, or an individual sound in an audible passage. In its general form, anchors may be associated with both the source and the target of a one-directional link in a hypertext. They present the relationship between the identified portion of the source and the identified portion of the target, a relationship more semantically ‘fine-grained’ than that expressed by pairing the entire content elements. In other implementations, anchors are only associated with source, with the target being the node as a whole.

In some system and data models, anchors have been identified as distinct components of a hypertext; see, for example, the Dexter reference model [18]. In our Trellis implementations, anchors may or may not be associated with the source—when no anchor is associated with a link source then the link is represented by a (graphical) button in a separately displayed palette.

Exactly what an anchor is, whether it is a distinct object or not, what operations can be performed on it—these are decisions made in constructing specific data models, that is, in the abstract structure of the r-model. As mentioned above, the r-model does not specify any particular approach. The *display* of anchors, however, within source and target contents is covered by the r-model in the mapping that defines the concrete content (specifically, in the concrete context level). Any mapping that implements the functionality of this level will specify both the form of the display and also anchor positions within whatever is

[16] See Leggett’s glossary [15], for example, for definitions of related terminology.

the appropriate content representation for source elements. Issues involving positioning of the target content's display when a link is followed are addressed in the definition of the concrete windows (specifically, the concrete hypertext level). This aspect is treated by the r-model as a viewport problem: how should a potentially large text block be clipped for viewing? If a target anchor is present, say near the end of the target element, then the concrete window for it would be positioned as a viewport on the end of the element's representation.

4.1.3 Different flavors of links

A hypertext implementation may contain several different kinds of links, each with a different implemented action on selection. The distinction between the different types of link is reflected in the r-model by a difference between the types of their corresponding abstract buttons.

The display of the source or target of a link may be static or may be computed. Such displays are described within the mapping that produces the concrete content representation.

In some circumstances selection of a link may cause an apparent change to the displayed content, for example, insertion of the target's content in place directly into the source. When the content actually changes in form, this is a matter of interest in the concrete content. However, when the content is actually unchanged in form, as is the case when the target material is inserted, this can be described through the display mapping that produces the concrete windows representation.

4.1.4 Dynamic content

Abstract content may be statically defined or it may be dynamically changing. Obviously, hypertext systems should be able to deal with either. One of the important properties of the r-model is a separation of hypertext content from structure. Mappings at the concrete and visible levels are responsible for creating the look of intertwined information, and if these mappings are managed in ways that do *not* appear to fix anchors deeply into content, for example, then dynamic content can be managed with the same system architecture as handles static content.

Dynamic contents can be classified by the point during execution at which displayable information becomes available for display, and by the manner in which reader interaction is allowed. For example, in the αTrellis system three forms of dynamic content have been accounted for:

- *Computed content*: Execution of an algorithm to produce some text, or a picture. This product of computation is then displayed as static content. Examples of computed contents are running a formatter on a text block before displaying it, or executing an operating system command like Unix's `ls` to get a listing of files in a directory. In this form, information is not displayed until the algorithm has finished execution.
- *Interactive content*: Execution of an algorithm that requires user input to complete execution. During browsing, the content computation begins when a node is visited, and it ends when the node is left. Information is available for display constantly during execution. Examples include text editors and spreadsheet calculators.

- *Filtered computation*: This form of dynamic content involves continuous execution of an algorithm, presumably to process some data stream. Even when the node for the content is not under visitation, the algorithm computing the content is executing. When the node is visited, an appropriate 'snapshot' is generated of the current accumulated state of the computation. An example of this form of dynamic content is filtering a satellite transmission for messages of several types; when the node is visited, a bar chart is drawn showing how many instances of each type have been found. Another example is a multiline graph drawn to reflect the past, say, 15 minutes of system performance metrics being captured by an operating system.

It may at first appear that the popularly argued cards vs. scrolling issue is also one of dynamic content, but in the r-model it is not. In neither case is content actually changing, but instead only the view on that content. Thus the matter becomes one that is handled by the mapping at the concrete hypertext level. This distinction is discussed in more detail in Section 4.2.2.

4.2 Aspects orthogonal to the r-model

The r-model is centered around organizing, categorizing, and relating the components and functions of hypertext models. Consequently, there are implementation elements, as well as elements of some hypertext models, that are not treated as architecture issues, but as issues specific to particular data and system models. As such, they are not explicitly dealt with in the r-model. Several such issues are discussed in the following sections.

4.2.1 Hypertext browsing semantics

We have previously defined a hypertext system's *browsing semantics* [2] ,as the dynamic properties of a reader's experience when browsing a document; in other words, as the manner in which the information within the hypertext is to be visited and presented. In most cases, browsing semantics are specified by the code that implements the hypertext system. However, it is also possible to develop a hypertext model with variable browsing semantics; for example our Trellis hypertext model permits specification of the hypertext's browsing semantics [19].[17] Although specifiable browsing semantics are in some hypertext models, they are not in all, and so we have decided not to include them directly in the r-model.

Similarly, we have not included the hypertext's dynamic behavior in the r-model. By dynamic behavior, we mean those cases in which a hypertext system traverses the structure without intervention from the reader [20]. Dynamic behavior is distinct from dynamic content, however. As noted above, dynamic content *is* described within the model.

4.2.2 Characteristics of the content

Some hypertext systems may favor an organization in which the content is partitioned into small, single-thought units while others favor organizations in which the content is

[17] The behaviors associated with different link types are reflected by their browsing semantics. Consequently, variable browsing semantics are the implementation mechanism for user-defined link types, as well as other browsing behaviors.

treated as larger, somewhat self-contained documents, each with more than one thought or point. Frequently the former organization is termed *card-based*, while the latter is termed *scroll-based*, in reference to the solution commonly used to present larger units in windows. Such considerations are outside of the scope of the r-model. This issue is related to the data to be used, and so precipitates decisions in designing the data model (i.e., abstract structure and abstract content) for specific systems. In r-model terms, a scrolling environment is viewed as a dynamic mapping at the concrete hypertext level (concrete windows), such that viewports onto the concrete contents are altered in response to reader actions. Card-based systems would correspondingly require far less manipulation of the viewports.

4.2.3 Physical-level descriptions and interchange descriptions

If the structure of the implemented hypertext system closely parallels that of the r-model, it will certainly be necessary to define a storage format for those representations that are specified directly as well as a description of the mappings that produce the others. However, the specific design of such storage formats is outside of the scope of the r-model, as is the equally important design of formats designed to permit interchange between hypertext systems and installations.

4.2.4 User interfaces

Certainly to the reader of a hypertext, the most visible component of the system is its user interface. However, the user interface is also an element of the system not discussed in the r-model. We note that it is possible to associate many different styles of user interface with the same underlying hypertext model.

4.3 Intersection with existing standards

There are two points of intersection between the r-model and existing standards. The first, in the abstract component level, are the abstractions used to define the abstract content. An appropriate standard to consider for text, for example, would be SGML [21]. Similar utility could be made of standards to define graphical material as well as other content objects. It may be necessary, however, to augment these standard representations with additional information describing the potential interactions defined by the concrete–structure and button–structure associations, and as reflected in the concrete content.

The other point of intersection with proposed standards is in the visible hypertext level. Each visible HT segment and user display may be based around a protocol such as that of the X-windows system [22]. Other de facto interface standards such as SunTools, OpenLook, Viewpoint, Motif, and NextStep are also applicable at this point.

5 DISCUSSION AND CONCLUSIONS

There are many other hypertext systems that have not been discussed in detail in this paper. We now briefly return to specific implementations in order to mention a few interesting and important features and describe how the r-model deals with them. All the systems discussed in Section 3 have been card-based, but this is not to imply that scrolling systems are inherently different or inappropriate for organization via the r-model.

(One well-known scrolling system is Intermedia [23,24].) As mentioned in Section 4.2 on orthogonal considerations, scrolling would be handled with viewport mapping. The remaining structure of these systems follows the r-model in much the same way as the other (card-based) systems we discussed.

Some systems predefine a rich set of link types, that is, specific and differing behaviors and presentations when links are followed. For example, Guide [25,26,27] defines a small set of types: replacement, note, and glossary buttons. Intermedia as well provides several different classifications, which Meyrowitz has generalized into generic behaviors called navigational links, warm links, and hot links [28,24]. Other systems, such as Trellis and HyperCard, contain a set of base link types and provide compositional facilities permitting authors to implement new link types from the base types. We discussed the r-model characterization of link types in Section 4.1.

A recent topic of discussion, termed *active anchors* [24], refers to how link terminals are specified and represented when content is especially dynamic, as in audio passages, video, or animation. Kacmar [29] has designed and implemented a system that uses an object-oriented network of actors exchanging messages. It incorporates active anchors through the actor mechanism. Another form of dynamic behavior appears in systems like Zellweger's scripted documents [30,31,32], that are intended to be used in creating structured presentations. Dynamic traversal of the resulting structures is an important part of achieving the purposes of these systems. The HyTime and SMDL standardization efforts [33,34,35] are also cast in a highly temporal environment and contain explicit mechanisms for describing duration and synchronization of events. In r-model terms, time is handled partially as structure and partially as (dynamic) content. Trellis mirrors the r-model in this respect, and separates the definition of dynamic displays (content) from automatic traversals of the links structure (timer-based events).

Several other research efforts have been aimed at producing an abstract model of hypertext and hypertext systems. Two that have been developed recently are the Dexter model [18] and one proposed by Lange [36]. Both of these models are more operational in nature than the r-model described here. These other abstractions are more directly useful as high-level specifications for system implementations, while the r-model is intended to be taxonomic in nature. As such, the r-model does not contain operational concepts, or specific data or presentation models. Rather than being an abstract engine or machine, the r-model has a functional organization showing relations among elements in the various entity levels.

In conclusion, we have described a meta-model of hypertext, which we call the r-model, that provides a high-level organization to the portions of a hypertext model or system. In this paper, we have focused on using the r-model to describe existing systems. We have seen that the detailed architecture of a system design may correspond closely to the divisions established in the r-model, as in the Trellis systems. However, it is equally possible that a system's design may merge distinct concepts in the r-model, such as in Hyperties and HyperCard.

Since the functionality identified by the r-model is inherent in hypertext applications, the r-model also can serve as a high-level design guide for implementors of systems. In our own work in developing the Trellis hypertext model and prototype implementations, we have tended to reflect the divisions of the r-model strongly in our hypertext model and also to carry these divisions on into our implementation. In essence, our implementation is based on a collection of abstract data types, where the data types correspond to the representations in the r-model. A natural consequence of this retention of separation has been that it is easy to extend the environment in which the implementation resides—for example to consider designs that permit multiple readers to be active in the hypertext at the same time that a writer is modifying it. Moreover the retention of separation between structure, content, and context permits flexible reuse of the hypertext's structure and of the content of the hypertext.

ACKNOWLEDGMENTS

We would like to thank the NIST Hypertext Standardization Workshop program committee, particularly Judi Moline, for comments that helped us to clarify the points of this paper. We also would like to thank the participants in the Workshop's Hypertext Reference Model working group, and especially John Leggett, for discussions that helped identify the similarities and differences between this model and the others that have been proposed.

REFERENCES

1. Richard Furuta and P. David Stotts, 'The Trellis hypertext reference model', in *Proceedings of the Hypertext Standardization Workshop*, eds., Judi Moline, Dan Benigni, and Jean Baronas, pp. 83–93. National Institute of Standards and Technology, (February 1990). NIST Special Publication 500-178. Workshop held January 16–18, 1990.
2. P. David Stotts and Richard Furuta, 'Petri-net-based hypertext: Document structure with browsing semantics', *ACM Transactions on Information Systems*, **7**(1), 3–29, (January 1989).
3. Alan C. Shaw, 'A model for document preparation systems', Technical Report 80-04-02, University of Washington, Department of Computer Science, Seattle, WA, (April 1980).
4. Richard Furuta and P. David Stotts, 'Separating hypertext content from structure in Trellis', in *Proceedings of Hypertext 2*, (June 1989). University of York, June 29th and 30th, 1989.
5. P. David Stotts and Richard Furuta, 'Temporal hyperprogramming', *Journal of Visual Languages and Computing*, (1990). **1(3)**, 237–253, (September 1990).
6. Cognetics Corporation, *Hyperties Author's Guide*, August 1988. Corresponds to version 2.3 of the software.
7. Richard Furuta, Catherine Plaisant, and Ben Shneiderman, 'Automatically transforming linear documents into hypertext', *Electronic Publishing: Origination, Dissemination, and Design*, **2**(4), 211–229, (December 1990).
8. Ben Shneiderman, 'User interface design for the Hyperties electronic encyclopedia', in *Proceedings of Hypertext '87*, pp. 189–194, (November 1987). Published by the Association for Computing Machinery, 1989.
9. Jacob I. Lifshitz, *Window Control Strategies for Hypertext Traversal*, Master's thesis, University of Maryland, Department of Computer Science, College Park, MD, 1989.
10. Apple Computer, Inc., *HyperCard User's Guide*, Apple Computer, Inc., 1987.
11. Apple Computer, Inc., *HyperCard Script Language Guide: The HyperTalk Language*, Addison-Wesley, 1988.
12. Judi Moline, 'The user interface: A hypertext model linking art objects and related information', in *Interfaces for Information Retrieval*, Greenwood Publishing Group, (1990). ASIS Monograph. To appear.
13. Judi Moline, 'Linking information to objects: A hypertext prototype for numismatists', *Visual Resources*, (1990). **7**, 361–377 (1990).
14. A. K. Dewdney, 'Wallpaper for the mind', *Scientific American*, 14–17, (September 1986).
15. John Leggett, John L. Schnase, and Charles J. Kacmar, 'Working definitions of hypertext', Technical Report TAMU 88-020, Texas A&M University, Department of Computer Science, (October 1988).
16. Brad Campbell and Joseph M. Goodman, 'HAM: A general purpose hypertext abstract machine', *Communications of the ACM*, **31**(7), 856–861, (July 1988).

17. Frank Wm. Tompa, ‘A data model for flexible hypertext database systems’, *ACM Transactions on Information Systems*, **7**(1), 85–100, (January 1989).
18. Frank Halasz and Mayer Schwartz, ‘The Dexter hypertext reference model’, in *Proceedings of the Hypertext Standardization Workshop*, eds., Judi Moline, Dan Benigni, and Jean Baronas, pp. 95–133. National Institute of Standards and Technology, (February 1990). NIST Special Publication 500-178. Workshop held January 16–18, 1990.
19. Richard Furuta and P. David Stotts, ‘Programmable browsing semantics in Trellis’, in *Hypertext ’89 Proceedings*, pp. 27–42. ACM, New York, (November 1989).
20. P. David Stotts and Richard Furuta, ‘Temporal hyperprogramming’, Technical Report CS-TR-2349 and UMIACS-TR-89-113, University of Maryland Department of Computer Science and Institute for Advanced Computer Studies, (November 1989).
21. ISO, *Text and Office Systems—Standard Generalized Markup Language*, October 1986. Document Number: ISO 8879–1986(E).
22. Robert W. Scheifler and Jim Gettys, ‘The X Window system’, *ACM Transactions on Graphics*, **5**(2), 79–109, (April 1986).
23. Nichole Yankelovich, Bernard J. Haan, Norman K. Meyrowitz, and Steven M. Drucker, ‘Intermedia: The concept and the construction of a seamless information environment’, *Computer*, **21**(1), 81–96, (January 1988).
24. Norman Meyrowitz, ‘The link to tomorrow’, *UNIX Review*, **6**(2), 58–67, (February 1990).
25. P. J. Brown, ‘Interactive documentation’, *Software—Practice and Experience*, **16**(3), 291–299, (March 1986).
26. P. J. Brown, ‘A simple mechanism for authorship of dynamic documents’, in *Text Processing and Document Manipulation*, ed., J. C. van Vliet, 34–42, Cambridge University Press, (April 1986). Proceedings of the international conference, University of Nottingham, 14–16 April 1986.
27. P. J. Brown, ‘Turning ideas into products: The Guide system’, in *Proceedings of Hypertext ’87*, pp. 33–40, (November 1987). Published by the Association for Computing Machinery, 1989.
28. Norman Meyrowitz, ‘Hypertext—does it reduce cholesterol, too?’, IRIS Technical Report 89-9, Brown University, Institute for Research in Information and Scholarship, (November 1989). Transcript of Hypertext ’89 keynote address.
29. Charles John Kacmar, *PROXHY: A Process-Oriented Extensible Hypertext Architecture*, PhD thesis, Texas A&M University, 1990.
30. Polle T. Zellweger, ‘Directed paths through collections of multi-media documents’, in *Hypertext ’87*, (November 1987). Position paper.
31. Polle T. Zellweger, ‘Active paths through multimedia documents’, in *Document Manipulation and Typography*, ed., J. C. van Vliet, 19–34, Cambridge University Press, (April 1988). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice (France), April 20–22, 1988.
32. Polle T. Zellweger, ‘Scripted documents: A hypertext path mechanism’, in *Hypertext ’89 Proceedings*, pp. 1–26. ACM, New York, (November 1989).
33. Steven R. Newcomb, ‘Explanatory cover material for section 7.2 of X3V1.8M/SD-7, fifth draft’, in *Proceedings of the Hypertext Standardization Workshop*, eds., Judi Moline, Dan Benigni, and Jean Baronas, p. 179. National Institute of Standards and Technology, (February 1990). NIST Special Publication 500-178. Workshop held January 16–18, 1990.
34. Steven V. Bertsche, ‘X3V1.8M SD-6 HyperMedia/time-based document (HyTime) and Standard Music Description Language (SMDL): User needs and functional specification’, Technical Report SD-6, ANSI Project X3.542-D, (April 1990). Second Draft.
35. Charles F. Goldfarb, Steven R. Newcomb, Donald Sloan, and Alan D. Talbot, ‘X3V1.8M journal of development; ANSI project X3.542-D; Standard Music Description Language (SMDL)’, Technical Report SD-8, ANSI Project X3.542-D, (April 1990). Sixth Draft.
36. Danny B. Lange, ‘A formal model of hypertext’, in *Proceedings of the Hypertext Standardization Workshop*, eds., Judi Moline, Dan Benigni, and Jean Baronas, pp. 145–166. National Institute of Standards and Technology, (February 1990). NIST Special Publication 500-178. Workshop held January 16–18, 1990.