Routes Through the Maze*

Olaf Weber†

September 4, 1995

Abstract

"You are in a maze of twisty little passages, all different."

Advocates of hypertext often refer to its 'structure' and 'non-linearity', as opposed to the lack of these qualities in 'normal' printed matter. Yet there is a lot of carefully thought-out structure in books, precisely because of the linearity of the medium, and it is frightfully easy to produce an amorphous labyrinth of nodes in a hypertext system.

This paper explores *routes* and *views* as ways of making it easier for writers to put structure into hypertexts, and for readers to use that structure effectively.

* *Financed by STW, project CWI22.2714*

† CWI, Amsterdam

Introduction

When reading the literature on hypertext, you may encounter the claim that it can be contrasted to texts on paper by its structure. This is at best a half-truth. There is a lot of structure in traditional texts, and many typographical devices are used to give a reader insight into that structure. On the other hand, a lack of structural clues in a hypertext can easily reduce it to the proverbial "maze of twisty little passages, all different".

The ACELA project is to deliver a viewer/editor for interactive (text)books, with a text on Lie algebras as a first example [1]. Since an interactive book is a kind of hypertext we are interested in presenting a text that allows readers the benefits of using its inherent structure without their getting lost.

The first section of this paper explores some of the means used to convey the structure of a printed text to its readers. Special attention is given to two cases where the text has a non-linear structure, as they are of prime interest to designers of hypertext.

The second section introduces the concepts of views and routes, as ways to describe the kinds of structuring that is needed in a hypertext.

The third section shows how routes and views can be modeled in terms of the nodes and links that make up a hypertext.

The fourth section contains some suggestions on what features an effective user interface to routes and views should offer.

Routes and views can help to make reading and writing a hypertext easier. The bottom line remains however that the freedom they offer can be abused as well as used: writing well is an art, for the computer just as much as for paper.

1 Paper

Over centuries of experimentation, the writers of paper documents have come up with a number of rules of thumb regarding how a document should be structured, and the typesetters have a corresponding set of rules regarding how this structure is to be presented to the readers of printed matter.

Some of these rules are obvious: the information that a writer wishes to convey often has a hierarchical structure, and the division of the document into chapters, sections, paragraphs and so on should mirror this structure.

In a similar vein, important things are typeset in large fonts and placed prominently on a page, while unimportant things are set in smaller fonts and may be tucked away in some unobtrusive corner. In the latter case, the main text contains only a small indication where that item can be found.

One aspect of the rules that make navigating a book easy is that it is obvious where to find things: tables of contents come near the front, appendices, bibliographies and indices at the end, in that order. And for detailed navigation, there's always the map of the book: the table of contents.

One aspect of paper documents is their fundamental linearity: most of them are intended to be read from start to end, with only trivial digressions. Sometimes writers are not content with this, and produce a book that looks very much like the printout of a hypertext. Encyclopedias and dictionaries are common examples. There are more interesting examples however, and we'll take a detailed look at two of them.

1.1 Textbooks

Our primary interest is textbooks, not literature. Common usage of a textbook is to read it once or twice, and then use if during an extended period as a reference. This dual purpose can strongly influence the structure of book that attempts to cater to both.

An example of a traditional approach to combining a tutorial and a reference manual is "LaTeX: A Document Preparation System", by Leslie Lamport [10]. This book is really a tutorial, with the reference manual tacked on as a (large) appendix. A complaint about the book is that the reference manual is complete, but lacks necessary explanations, while the tutorial has the explanations, but isn't complete. This makes the book hard to use in either role. (The second edition [11] is an improvement, but still not ideal.)

1.2 Dangerous Bends

The first example of techniques normally associated with hypertext is found in "The TeXbook" [8] and "The METAFONTbook" [9], both by Donald E. Knuth. The TeXbook (referred to as the *TeX book* in the rest of this text) introduces the TeX typesetting system. The book tries to cater to experienced users and novices alike, and its structure reflects this.

Traditionally, such books are written with the difficulty gradually increasing in later chapters. However, that approach can lead to a fragmentation of material that logically belongs together.

Knuth used a different approach. Parts of the text are marked by 'dangerous bend' symbols to indicate that the material is relevant for experienced users only. Other parts are marked by a 'double dangerous bend'. These are only to be read by those people who want to know *everything.*

Also, the organization is such that the 'unmarked' text may be inaccurate or incomplete, with the details needed to cope with obscure situations shielded by the bends, or supplied in later chapters.

For example, here is some text from page 14 that conveys a bit of the flavour of the system.

> Old manuals of style say that the punctuation after a word should be in the *same* font as the *word;* but an italic semicolon often looks wrong, so this convention is changing. When an italicized word occurs just before a semicolon, the author recommends typing '`{\it word\/};`'.
> Every letter of every font has an italic correction, which you can bring to life by typing `\/`. The correction is usually zero in unslanted styles, but there are exceptions: To typeset a bold '**f**' in quotes you should say `a bold '{\bf f\/}'`, lest you get a bold '**f**'.
> The primitive control sequence `\nullfont` stands for a font that has no characters. This font is always present, in case you haven't specified any others.
> Fonts vary in size as well as in shape. For example, the font you are now reading [. . .]

All this invites a strategy of reading the book thrice, picking up the bends on the second reading, and the double bends on the third. Appendix D ("Dirty Tricks", with *eleven* dangerous bends) could be read some time after that.

For a tutorial, this strategy works very well, but what happens when the TeX book is used as a reference? Knuth supplies the readers with an extensive index, and typographic clues indicate which references are the best to follow.

However, while the structure of the TeX book does helps to keep information together, the global (tutorial) structure of the book still results in a separation of items that should come together in a reference manual. For some items, the most important information alone is spread over 5 chunks. As a result, the TeX book is more suited for use as a tutorial than as a reference manual.

1.3 A Matrix of Contents

The second example of the use of hypertextoid techniques in paper books is "Informal Introduction to ALGOL 68, Revised Edition", by C. H. Lindsey and S. G. van der Meulen [14]. This book (known as the *ALGOL book* in the rest of this paper) was written as a more accessible companion volume to the "Revised Report on the Algorithmic Language ALGOL 68", by Van Wijngaarden et al. [17]. ALGOL 68 introduced (among others) the concept of orthogonality into programming languages. The language was designed by providing a few basic concepts, which can be combined freely. The book is structured to reflect this notion.

This is achieved by having eight *horizontal* chapters, and seven *vertical* chapters. The horizontal chapters are subdivided into seven sections, and the vertical into eight sections. The net

	x.1 **FUNDAMENTALS**	*x*.2 **PROCEDURES AND NAMES**	*x*.3 **OPERATIONS**
1.*y* **BASIC CONCEPTS**	1.1 FUNDAMENTALS 1.1.1 Objects 1.1.2 Identifiers 1.1.3 Phrases, serial and collateral elaboration 1.1.4 Routines 1.1.5 Defining and applied occurences 1.1.6 Coercion	1.2 NAMES AND DECLARERS 1.2.1 Ascription and assignation 1.2.2 Identity declarations 1.2.3 The metanotation MODE 1.2.4 Summary	1.3 SYMBOLS, MODES AND OPERATORS 1.3.1 Representations 1.3.2 Symbols, bold words and comments 1.3.3 Other declarations
2.*y* **DECLARATIONS**	2.1 PRIMITIVE DECLARATIONS 2.1.1 Primitives 2.1.2 Variable declarations 2.1.3 Sample declarations	2.2 IDENTITY DECLARATIONS 2.2.1 Identity declarations 2.2.2 Another look at variable declarations 2.2.3 Initialised variable declarations	2.3 MODE DECLARATIONS
3.*y* **CLAUSES**	3.1 SERIAL CLAUSES 3.1.1 The declarations 3.1.2 The statements 3.1.3 The yield 3.1.4 Completers 3.1.5 Delimiters	3.2 CLOSED CLAUSES 3.2.1 Ranges and reaches 3.2.2 Scopes of names 3.2.3 Identification 3.2.4 ENCLOSED clauses	3.3 BOLD WORDS 3.3.1 Identification of mode-indications
4.*y* **ROUTINES**	4.1 PROCEDURES AND OPERATORS 4.1.1 Standard prelude routines	4.2 PROCEDURE DECLARATIONS 4.2.1 **proc** declarers 4.2.2 Routines 4.2.3 Scopes of routines	4.3 OPERATION DECLARATIONS 4.3.1 Priority declarations 4.3.2 Operation declarations 4.3.3 Identification of operators
5.*y* **UNITS** 5.*y*.0 Coercion 5.*y*.1 Primaries 5.*y*.2 Secondaries 5.*y*.3 Tertiaries 5.*y*.4 Quarternaries	5.1 SIMPLE UNITS 5.1.0 Coercion 5.1.1 Primaries 5.1.2 Secondaries 5.1.3 Formulas 5.1.4 Quarternaries	5.2 BALANCE AND CALL 5.2.0 Coercion 5.2.1 Procedure calls 5.2.3 **nil** 5.2.4 Assignations involving names	
6.*y* **STANDARD PRELUDE**	6.1 OPERATORS 6.1.1 Monadic operators 6.1.2 Dyadic operators	6.2 CONSTANTS AND PROCEDURES 6.2.1 Constants 6.2.2 Procedures	6.3 ASSIGNING OPERATORS
7.*y* **TRANSPUT**	7.1 FORMATLESS TRANSPUT 7.1.1 Formatless output 7.1.2 Formatless input	7.2 FILES 7.2.1 Channels, books and files 7.2.2 Environment enquiries 7.2.3 Procedures for opening and closing 7.2.4 Position enquiries 7.2.5 Layout routines	
8.*y* **EXAMPLES**	8.1 SIMPLE EXAMPLES	8.2 PROCEDURE EXAMPLES 8.2.1 Easter	8.3 EXAMPLES OF OPERATORS 8.3.2 Parallel plus

The matrix of contents

result resembles a two-dimensional matrix, which is what the table of contents looks like. A look at the figure reveals two interesting points: the structure is only one level deep, as each section is itself a linear text, and there are a number 'holes' in the matrix where no text is provided at all.

For example, the figure shows that horizontal chapter 4 covers "Routines", and vertical chapter 2 "Procedures and Names". Thus the intersection of these two is about "Procedure Declarations", and functions as both the second section of the fourth horizontal chapter, and the fourth section of the second vertical chapter.

Not shown in the figure is that the main text of the book is preceded by "A Very Informal Introduction to ALGOL 68" (which serves to bootstrap the reader), and followed by some appendices.

At the end of the introduction, the authors remark:

> The [vertical] route is the more didactic one, for those who wish to learn the language. The horizontal one [the way the book is bound] is more appropriate for those who wish to survey the essential principles of the language as a whole. [. . .]
>
> If you are now in some doubt as to which route is for you, then take our suggestion – read horizontally in Chapter 1 until you find the waters beginning to get a little deep: then return to 2.1 and read by the vertical route thereafter.

x.4 STRUCTURES	**x.5 MULTIPLE VALUES**	**x.6 UNIONS**	**x.7 DISTINCTIVE FEATURES**
1.4 STOWED VALUES, STRUCTURES 1.4.0 STOWED values 1.4.1 Enumeration by tagging 1.4.2 Different objects in one box 1.4.3 Chaining 1.4.4 Pandora's boxes	1.5 STOWED VALUES, MULTIPLES 1.5.1 Multiple values and descriptors 1.5.2 Indexing 1.5.3 Identifier declarations for multiples 1.5.4 Slices 1.5.5 Interrogations	1.6 UNIONS 1.6.1 United modes 1.6.2 Assignations and conformity-clauses	1.7 DISTINCTIVE FEATURES 1.7.1 The **long** and **short** modes 1.7.2 Identity relations
2.4 STRUCTURE DECLARATIONS 2.4.1 **struct** declarers 2.4.2 **struct** declarations 2.4.3 Well formed modes 2.4.4 The mode **compl**	2.5 MULTIPLE DECLARATIONS 2.5.1 Row declarers 2.5.2 Row declarations 2.5.3 The mode **string**	2.6 UNION DECLARATIONS 2.6.1 **union** declarers 2.6.2 **union** declarations	2.7 BITS, BYTES, LONGS AND SHORTS 2.7.1 **bits** and **bytes** 2.7.2 **long** and **short** modes 2.7.3 **heap** declarations
3.4 STRUCTURE DISPLAYS	3.5 ROW DISPLAYS AND LOOPS 3.5.1 Row displays 3.5.2 Loop clauses	3.6 CONFORMITY CLAUSES	3.7 COLLATERALITY 3.7.1 Collateral clauses 3.7.2 Parallel clauses
	4.5 ROW-OF PARAMETERS		4.7 JUMPS 4.7.1 Simple jumps 4.7.2 Procedured jumps
5.4 UNITS AND STRUCTURES 5.4.0 Complex widening 5.4.1 Applied identifiers 5.4.2 Selections 5.4.3 Formulas with complex operators	5.5 UNITS AND MULTIPLES 5.5.0 Rowing 5.5.1 Primaries 5.5.2 Multiple selections 5.5.3 Bound interrogations 5.5.4 Assignations	5.6 UNITS AND UNIONS 5.6.0 Uniting 5.6.1 The **void** denotation 5.6.4 Assignation of unions of rows	5.7 BITS AND PIECES OF GARBAGE 5.7.0 Coercion 5.7.1 Primaries 5.7.2 Generators 5.7.3 Order of elaboration of operands 5.7.4 Identity relations
	6.5 INTERROGATIONS 6.5.1 Dyadic operators 6.5.2 Monadic operators		6.7 LONG OPERATORS 6.7.1 Environment enquiries 6.7.2 Procedures 6.7.3 Operators 6.7.4 **leng** and **shorten** 6.7.5 **up** and **down**
7.4 STRUCTURES AND EVENTS 7.4.1 Straightening of structures 7.4.2 Files 7.4.3 Code conversion 7.4.4 Event routines	7.5 ROWS AND STRINGS 7.5.1 Straightening of multiple values 7.5.2 Conversion procedures 7.5.3 Conversion environment enquiries	7.6 FORMATTED TRANSPUT 7.6.1 **format** texts 7.6.2 **format**s 7.6.3 The formatted transput procedures 7.6.4 Events	7.7 BINARY TRANSPUT 7.7.1 Binary transput procedures 7.7.2 Some restrictions
8.4 TWO EXAMPLES OF LIBRARY PRELUDES 8.4.1 Operations on vectors in E_3 8.4.2 Operations on rational operands	8.5 A LIBRARY PRELUDE FOR VECTOR AND MATRIX OPERATIONS IN E_n 8.5.1 Operations on vectors in E_n 8.5.2 Operations on matrices and vectors 8.5.3 Operations on square matrices	8.6 EXAMPLES OF TRANSPUT 8.6.1 The happy family	8.7 EXAMPLES OF EVERYTHING 8.7.1 Analytic differentiation

of the ALGOL book.

This sounds as if the authors would have structured the book with horizontal chapter one first, followed by the vertical chapters (minus their first sections), had they been more traditionally minded.

It is interesting to contrast this approach with the one taken by Lamport (section 1.1). In the ALGOL book the reference manual text and the tutorial text are one and the same. Thus, any lacunae in the tutorial are avoided. Moreover, the reference text reads, necessarily, as a part of a tutorial. It is almost the best of both worlds.

One question remains though: the 'default route' through the ALGOL book corresponds to the reference manual, not to the tutorial. Should this not have been the other way round? As the 'reference phase' of a textbook's life-cycle lasts much longer than the 'tutorial phase', it is probably better to obtain a reference structure at the expense of the tutorial structure than vice versa.

Still, it would be nice if we could avoid making this choice, and using hypertext may give us this option.

2 Views and Routes

In the ALGOL and the TEX book we see two mechanisms that can be exploited to structure hypertexts. The ALGOL book demonstrates that the parts of a text can be organized into different *routes* that a reader can follow. The TEX book shows that it may be useful to present different *views* of parts of a text to different readers. Combining these mechanisms makes it possible to exercise a fine-grained control over what a user sees of a text.

The following sections treat possible uses of views and routes in more detail.

2.1 Views

A *view* is a way of presenting (a part of) a text to user. There are a number of reasons why it is useful to able to present different views, and we will concentrate on three of them. Other examples can be found in [1].

Giving a reader insight in the global structure of a text is one common reason. On paper this function is performed by the table of contents. Some books have two tables of contents: one providing only chapter names, the other giving a fine-grained breakdown of each chapter. A common problem with such books is that the former gives too little information, but the latter too much.

Working on a computer allows us to strike a balance between these extremes by providing a table of contents in which only the current chapter is described in detail. This is an example of a 'fish-eye view' as described by Furnas in [4], and is similar to functionality provided by outline editors.

Other ways to provide similar information is by giving a two-dimensional 'map', and allow a reader to determine the desired level of detail by a kind of 'zooming' mechanism.

The second reason is exemplified by the TEX book. Not all the material in a piece of text will be useful for every reader. The beginners view of the quote given could be

> Old manuals of style say that the punctuation after a word should be in the *same* font as the *word;* but an italic semicolon often looks wrong, so this convention is changing. When an italicized word occurs just before a semicolon, the author recommends typing '`{\it word\/};`'.
>
> Fonts vary in size as well as in shape. For example, the font you are now reading [. . .]

although it is probably better to leave an indication that text for more advanced users has been left out.[3]

A third reason is exemplified by Knuth's WEB system [7]. A WEB source text has two other views: a typeset document describing a Pascal program, and the program code itself. The latter view is meant to be used by the compiler, and quite hard to read by humans as no attempt at pretty-printing is made. In the ACELA project we have a similar situation with mathematical objects, see [13] for details.

2.2 Routes

A route is a linearization of (a part of) a hypertext. The purpose of a route is to put the parts of a text into an order that can be comfortably read from begin to end, and thus relieve users from the problem of finding their way around all by themselves.

On paper, there is usually a single route through the text, and part of the art of writing well is to make sure that this route is as effective as possible at conveying the writer's thoughts.

In a hypertext it may still be useful to have a route, prepared by the writer, that will take the reader 'from cover to cover'. The reader who follows that route won't have to remember every place where the path forked, and isn't at the end left with the nagging suspicion that important parts might have been missed.[1]

When more than one route makes sense, as in the ALGOL book, the writer can supply several routes, and readers can change their current route to another one. The ability to change the current route becomes really useful when the new route was generated at the reader's request.

There are several conceivable ways of generating new routes. For example, a reader of the ALGOL book might be interested in "variable declarations". These are covered by sections 2.1.2, 2.2.2 and 2.2.3 in the book. In order to understand these sections, the reader should probably read 2.1 and 2.2 entirely, and 1.1 and 1.2 because the interesting parts depend on them.

The general mechanism here is that there is dependency relation between the parts of the text, and this can used to obtain a short-list that covers all required reading for the matter of interest. This 'read before' relation is often not sufficient to fix a unique route: in the example, there are two routes that satisfy the 'read before' constraint: 1.1 – 2.1 – 1.2 – 2.2 and 1.1 – 1.2 – 2.1 – 2.2.

This is again a point where a cover-to-cover default can be useful. If the reader is currently following the 'tutorial route', it makes sense to use retain that order in the nodes to be read. Thus the latter option would be suggested.

This facility cannot replace writer-supplied routes. It is quite possible that a generated route to the 'last' node of the text skips large parts of it, or puts large sections in a random order.

One common way to obtain a route is as an excursion from the main text. The simplest example of this is a short footnote: readers go there, read it, and go back to where they came from. This is an example of a general pattern: the *digression*.

A digression occurs when a reader takes a (presumably) brief interest in a tangential matter, follows a route that provides the desired information, and continues from the point of departure.

Digressions can be arbitrarily complex, and most people who have ever looked up something in an encyclopedia will know how easy it is to lose yourself in a succession of digressions. When the system keeps the 'point of departure' it becomes a lot easier to return to the matter at hand.

One source of digressions is when a text has been searched for a word, and the occurrences must be viewed. The result of the search can be presented as a route. The order of the parts would now not depend of the 'read before' relation, but (for example) on the number of times the search term occurs in that part of the text.

The linearity of routes has another consequence. It allows them to be presented as a kind of table of contents. This in turns allows the mechanisms used to view a table of contents to be used on any route at all. So we have a unified interface metaphor for several functions.

3 Nodes + Links = Hypertext

A hypertext consists of interconnected pieces of text called *nodes*. The relations between nodes are described by *links*. In this section, we take a closer look at how views and routes fit in such a nodes & links model.

3.1 Nodes

In our model, nodes are the units from which routes are constructed. This does not imply that a single node is just a piece of text with no further structure.

The beginner's route through a text, for example, might not only different nodes than the expert's route, the individual nodes may also look different, as with parts of the TEX book.

This aspect is modeled by modelling a node as an item of data to which *presentation rules* must be applied in order to generate a view of it. Different sets of presentation rules define different

[1]Such a feeling might be a proper literary effect, but it is disastrous in a textbook.

views. In this model, a table of contents would be single node that can be viewed in a (large) number of ways.

3.2 Links

In our model, links go from a source to a target, but can be followed in reverse direction as well. One link might connect more than one source to several targets. The endpoints of a link are its *anchors*. An anchor can be an entire node, or a part of the contents of a node.

Links are used to express several kinds of relationships, and a number of taxonomies have been proposed, for example see [2]. For our purposes we need a taxonomy that contains the following elements:

Organizatorial Links These links describe the logical structure of the text. They can be divided into two categories.

> **Structural Links** These links describe how nodes are grouped into a hierarchic structure, the familiar division of a text into chapters and sections. Taken together they usually form a tree.
>
> **Sequential Links** These links represent the 'read before' relation, by referring from a source node to the nodes it depends on. Taken together they should form a directed acyclic graph.

Associative Links These links provide references to related, but tangential informations. They supply to the connection to a number of digressions, like notes or cross-references.

> **Annotational Links** These links lead to notes to the text. The destinations of these links are not a part of the main text, and cannot be reached through structural links.
>
> **Referential Links** These links provide for cross-references to other parts of the text.
>
> **External Links** These links are like referential links, except that their target is (a part of) a different text. It may be impossible to follow these links in the reverse direction.
>
> **Implicit Links** This is a catch-all category for links that are not explicitly stored: an attempt to follow them generates a query. A typical example is dictionary lookup. The reverse direction of a dictionary lookup would deliver a list of nodes in which a certain word occurs, which is virtually an index mechanism.

The difference between structural and sequential links is demonstrated by the ALGOL book: there is a sensible route in which section 2.2 follows 1.2, but comes before 2.1, and vice versa.

It is mainly from a sense of tidiness that annotational links are separated from structural links. The relationship of a (foot)note or annotation to the text it belongs to is quite different from the relationship between a section and the chapter it is a part of. Moreover, footnotes are sometimes shared, and thus do not conform to the tree-like structural links – but there are workarounds.

Referential links provide for references across the hierarchical structure formed by the structural links, allowing it to retain its tree-like quality.

The limitation on sequential links implies that relations like "it is sufficient to have read chapter 3 *or* chapter 4" cannot be expressed. This doesn't seem a real limitation, as such relations within a text are extremely rare.

On the other hand, it isn't very unusual for a text to specify that one of a number of other texts is sufficient background to start reading it. However, such enumerations are rarely complete when written, and are virtually impossible to keep up-to-date. There are also technical reasons to treat links to other texts differently: the material referenced need not be available electronically, and if it is will typically be found on someone else's machine.

4 Users

To close, some general thoughts on how the mechanisms might look to people using them. The contents of the sections below are suggestions for aspects of a user interface that makes using routes more effective. Some of these suggestions have been validated by research, others are just based on common sense.

The distinction between readers and writers is somewhat artificial: readers may not be allowed to change the main text, but their notes can be complete texts in their own right. Thus it makes sense to integrate editor and viewer in a single system. Nevertheless, the two modes of use are sufficiently different to warrant separate treatment.

4.1 Readers

The general idea behind the reader's interface is that almost all of it should be adjustable. The writer is responsible for supplying reasonable defaults to the reader.

When reading a document, one window will contain a view of the current route, while another contains the node that is being read. Having both available at the same time allows a reader to preserve a better overview of the text, and allows in many cases for a better use of the width of the computer screen. [12] [15]

A 'go forward' command scrolls the text in the window and replaces it with the next node when necessary. This is similar to the way the space-bar works in many news-readers.

When deviating from the route (by following a non-structural link), a new window pops up. The 'forward' action then leads you through the excursion. At its end the window disappears again, and you're back at the point where the diversion began. Keeping the main text around when digressing serves both as a reminder of what you were doing, and allows for comparisons between the main text and the associated one.

The ability for readers to edit the way objects are viewed allows not only for changes in font sizes, but also regarding which notations are used for mathematical concepts. In the field of Lie algebras (for example) there are several incompatible notations for some concepts, and having to 'switch' between them can be a nuisance.

Readers should be allowed to create their own routes through a text, as well as annotate it. These features are be especially useful if these routes and annotations can be communicated to other readers.

4.2 Writers

The general idea behind the writer's interface is that virtually everything should be adjustable. The system should generate reasonable defaults, so that writers can concentrate on producing a text.

Constructing the main table of contents while writing makes the creation of sequential links easy: section 3.1 precedes 3.2, and chapter 3 precedes chapter 4. For the ALGOL book, generating the sequential links for the horizontal and vertical tables of contents along these lines gives exactly the desired pattern.

Of course, when special care is required, an editing tool for the sequential links is needed. It must answer questions about which nodes depend on each other, and whether circularities exist. While circularities should be avoided, it isn't that hard to determine reasonable routes in their presence. This means that there is no pressing need to do more than alert a writer to their presence, which is useful, as they might arise during editing, while being removed as a matter of course before the text is delivered.

The limitations on external links should not really be a hindrance. If two (or more) texts are sufficiently interrelated to warrant a complex web of links between them (as with a Bible book and commentaries), then they should probably be treated as a single text in our model. See also [15] and [5].

Writers need the ability to edit the way objects are viewed even more than readers do. Not only will they want to customize things like fonts shapes and sizes, they may have to create a large part of the notations they want to use from scratch. Designing a good interface for this is still an open problem.

5 Conclusion

We have seen reasons why there is a fundamental tension in textbooks between their dual roles as tutorials and reference works, and some of they ways used to get around this. This lead us to the concepts of a view of a text, and of a route through a text. And we have seen how they can be implemented as part of a hypertext system.

To readers of a hypertext views and routes offer a great improvement over paper. However, hypertexts have to be written before they can be read. Here too, views and routes can help writers of hypertext to produce a well-structured product.

It should be remembered though that while technology can enable good writing, it cannot enforce it. Constructing a (hyper)text that isn't a mere maze of words remains an art.

References

[1] Arjeh M. Cohen, Lambert Meertens (1994). The ACELA Project: Aims and Plans. *In print (HISC).*

[2] Steven J. DeRose (1989). Expanding the Notion of Links. *Hypertext '89 Proceedings.*

[3] Shelley Evenson, John Rheinfrank, Wendie Wulf (1989). Towards a Design Language for Representing Hypermedia Cues. *Hypertext '89 Proceedings*, 83–92.

[4] George W. Furnas (1986). Generalized Fisheye Views. *CHI '86 Proceedings*, 16–23.

[5] Robert J. Glushko (1989). Design Issues for Multi-Document Hypertexts. *Hypertext '89 Proceedings*, 51–60.

[6] C. McKnight, A. Dillon, J. Richardson (1993). Hypertext, a Psychological Perspective. *Ellis Horwood, New York, London, Toronto, Sydney, Tokyo, Singapore.*

[7] Donald E. Knuth (1983). The WEB System of Structured Documentation. *Stanford University, Computer Science Department, Report R 83/980.*

[8] Donald E. Knuth (1984). The TEXbook. *Addison-Wesley, Reading, Masschusetts.*

[9] Donald E. Knuth (1984). The METAFONTbook. *Addison-Wesley, Reading, Masschusetts.*

[10] Leslie Lamport (1985). LATEX, A Document Preparation System. *Addison-Wesley, Reading, Massachusetts.*

[11] Leslie Lamport (1994). LATEX, A Document Preparation System, Second Edition. *Addison-Wesley, Reading, Massachusetts.*

[12] Thomas Landauer, Dennis Egan, Joel Remde, Micheal Lesk, Carol Lochbaum, Daniel Ketchum (1993). Enhancing teh Usability of Text Through Computer Delivery and Formative Evaluation: the SuperBook Project. *[6], 72–136.*

[13] André van Leeuwen (1994). Representation of Mathematical Objects in Interactive Books. *In print (HISC).*

[14] C. H. Lindsey, S. G. van der Meulen (1977). Informal Introduction to ALGOL 68. *North-Holland, Amsterdam.*

[15] Christine M. Neuwirth, Ravinder Chandhok, David S. Kaufer, James H. Morris, Paul Erion, Dale Miller (1995). Annotations are not "for free": The Need for Runtime Layer Support in Hypertext Engines. *[16], 156–166.*

[16] W. Schuler, J. Hannemann, N. Streitz (Eds.) (1995). Designing User Interfaces for Hypermedia. Research Reports ESPRIT, Project 6532, HIFI, Volume 1. *Springer-Verlag, Berlin, Heidelberg, New York.*

[17] A. van Wijngaarden, B. J. Mailloux, J. E. .L. Pek, C. .H. .A. Koster, M. Sintzoff, C. H. Lindsey, L. G. L. T. Meertens, R. G. Fisker (1976). Revised Report on the Algorithmic Language ALGOL 68. *Mathematisch Centrum, Amsterdam.*