Creating Multimedia Documents: Hypertext-Processing

Sebastian Rahtz, Les Carr and Wendy Hall
Department of Electronics and Computer Science
University
Southampton S09 5NH
e-mail: spqr@uk.ac.soton.ecs

April 26th 1989

Abstract

The concepts of hypertext and hypermedia have created a flurry of activity both inside and outside the world of computer science but much of the work undertaken so far has been concerned with creating hypertext interfaces that allow an author to create documents in a hypertext format, incorporating buttons and links in the ways permitted by the particular hypertext system. This work is of fundamental importance to our understanding of hypertext systems and the nature of the structures they allow us to create. It is also vital, however, that hypertext systems provide interfaces to existing text-processing systems so that documents created in this way can easily and perhaps automatically be incorporated into a hypertext format. At the same time, this would provide a hypertext interface for viewing, accessing and cross-referencing such documents.

This paper describes a project being undertaken in Southampton to provide a hypertext interface for texts written using a generic markup language, such as LaTeX. The author creates an ordinary LaTeX document using normal facilities for cross-referencing, indexing etc., and this generic information is used to generate tags for a hypertext interface, which allows the document to be viewed section by section and provides buttons for accessing tables, diagrams, figures, references and cross-references to other documents.

1 Introduction

Much attention has been paid to the use of hypertext techniques in presenting material of a type unsuitable for formal databases, providing hybrid 'documents' which can be perused in a non-linear fashion. Much of the effort (and the hype) has gone into developing systems which permit interactive linking of items, arbitrary browsing and access to multiple media sources; the emphasis has been on the need for the reader to meander *ad lib.* through a world of pure 'knowledge' (cf Bush, 1945, Nelson, 1974, Nelson, 1967, **?**, Yankelovich et al., 1988), adding to it *en route*. This rather naïve, one-dimensional, view of 'knowledge' has seemed opposed to the quite different world of document markup and typesetting, where the emphasis is on fully-authored documents which need to be understood by a wide range of formatting software designed to create printed matter, using the appropriate layout tools to instantiate authorial effects (cf Bryant, 1988). This is not to deny the excellent hypertext work being done with large, formally structured, databases (cf Akscyn et al., 1988, Akscyn and McCracken, 1984, **?**), rather to point out the general conception of hypertext as a 'free' medium (an impression due in no small degree to the design of Hypercard). Our aim here is not only to try and show how a single document can be written for processing both by traditional formatting systems and by a hypertextual front-end, but also to demonstrate how multi-media references can be seamlessly integrated with pure text in a way previously associated with traditional hypertext.

What do we expect from a hypertext system? Conklin (Conklin, 1987) gave 12 characteristics by which to judge a hypertext system; but are these *characteristics* actually *desiderata*? Conklin only gives passing mention to the editing tool used to create new nodes, and the assumption is usually that the magic system chosen is to be the final resting place of the data, and although tools are provided to import text and pictures, little effort is expended on tools to write *out* a generalised version of the knowledge suitable for other systems. Import and export facilities generally relate to the low-level elements of the document (words, numbers, bitmaps), and a general criticism can be made that hypertext shares with desktop publishing a view of documents as a continuous stream of words, which are broken up into pleasant-looking groups with no explicit decisions about what the groups represent. But there is also a place for *generic* creation of material for hypertextual access. Books, after all, have proved to be excellent tools for knowledge transference, and we should not lightly abandon the concept of linear argument and explicitly-authored hierarchies simply to use the new tools of free browsing and arbitrary links. Hypertext has tended to start afresh in knowledge representation (witness the medium-driven adoption of the screen-sized 'card' as a vehicle for representation of information—Guide (Brown, 1986, Brown, 1987, Tebbutt, 1988) tends towards being an exception), and practitioners have struggled to add on traditional book features (thus Conklin rightly lays stress on the need for 'maps' of the hypertext space, parallel to the traditional table of contents). The work outlined here starts from the opposite direction—taking material marked up for traditional publication and viewing it with hypertext tools.[1]

[1] The Lace system is primarily the work of Les Carr; the multi-media provision is the domain of Wendy Hall, and Sebastian

2 The Lace system

Lace is a Southampton hypertext front end to documents, running on a SUN workstation and consisting of the following components:

1. LaTeX, the author's medium for creating documents;
2. NeWS, the windowing display system used to show the elements of the documents, and provide the reader's interaction with the documents;
3. A document library server, which manages requests from the user and display system not only for text but also other media (pictures, music etc)

LaTeX was chosen as the standard authoring system since it is a *structured markup logical language*; troff has also been used (with various high-level macros), but has a disadvantageous input syntax and an inability to pass 'special' information unhindered through the formatting stage. It must be stressed, however, that our interest is in any structured markup, and our only reason for not using SGML was the widespread use of LaTeX and its excellent underlying formatting system, TeX. It is assumed for the purpose of this paper that the reader is broadly familiar with the use of such a system, in which the author explicitly declares structures like section hierarchies, lists, quotes, citations, cross-references, index entries etc, but does not concern him/herself with how they should be implemented in a given display medium (such as a book, an article, or a manual etc).

NeWS was chosen for the display system since it is network based, interpretive (and hence easily modifiable) and uses the well-known PostScript language for communication. This version of PostScript has been augmented to handle input, processes and events. Future versions of Lace will use X Windows and Sunview.

2.1 The Philosophy of Lace

The traditional view of hypertext is that the knowledge universe is composed of *nodes* of information which are joined by unidirectional *links*. Lace shares this view, but sees the nodes not as independent entities, rather grouped as part of a larger structure: the *document*. A node may be a chapter, section, table, footnote or any other document substructure. Information is addressable to the node level by giving the name of the document and the name of the node inside the document (*e.g.* `mypaper:subsection The Philosophy of Lace`). Lace therefore considers a document to be a structured collection of nodes in much the way that SGML tries to formalise.

Each document may be of a different *type*, implying a different structure (*e.g.* LaTeX's rich hierarchical structure compared with WEB's simple structure of sections composed of documentation and code fragments) or a potentially different medium (*e.g.* LaTeX as textual information compared with videodisc technology 'movie' documents). It is important to note that documents of different media also have structure. Somewhat akin to theatrical *acts*, videodiscs are broken up into *chapters*, whose informational content can be notionally further subdivided into *sections* and so on.

The document server keeps a database of locally maintained documents, holding the following information about each document: the full title, a shorter name for convenient reference, its location in the local filestore, keywords that classify it, the type of the document, and details of the groups of users who have permissions for access to it. Adding a document to this database consitutes 'publishing' the document, *i.e.* making it available to other users for browsing and to other documents for reference. The server also maintains a map of each document's structure which is created when the document is formatted. This allows requests for 'document:node' to be satisfied by one lookup in the host's document database and one lookup in the document's map.

2.2 Lace meets LaTeX

LaTeX provides various methods for presenting and referencing information. The following list shows how Lace has taken advantage of them.

cross references LaTeX has `label` and `ref` commands for *labelling* an item (such as a section) and making a reference to that item (with a piece of text like *see page 3*) at a later stage. We could consider this to be a

Rahtz has provided the example system used below.

hypertextual link *from* the reference text *to* the labelled section, and can be implemented by a button over the reference text which when pressed, causes page 3 to be displayed. The links may also be *defocused*, providing a means to jump to the higher-level section within which the direct links occurs—thus a cross-reference may point explicitly to some text in section 3.5.1, but we can offer a facility to go to the start of section 3.5, or section 3.

citations Citing other works is where hypertext really comes into its own. Lace allows two forms of citation, one where the document is published in Lace form, and a mouse-press on the citation mark will bring up a window displaying the cited work, and the second where the document is not available to Lace, where a window pops up with the corresponding bibliography entry.

table, figures, footnotes One of the common ideas of hypertext is that of a button which *hides* some information—similar in function to a fold in a folding editor. This concept can be used for showing tables, graphics and other display material which doesn't form part of the continuous flow of text. Displaying this type of material traditionally involves *floating* it to a new page, perhaps collecting several display items to be placed on a page by themselves. These items are referred to by the label/reference method described above. Lace allows such elements to be removed from the main body of the document, and displayed in a separate window on request. That request usually takes the form of a button over a piece of reference text like "table 5 shows…".

sectioning A table of contents provides the reader with a *map* of the document's structure. As well as the traditional 'contents page' Lace implements this as a hierarchy of menus, separate from the actual text of the document.

index An index as well connects different parts of a document together—those parts refering to the same keyword. As well as a traditional index page with a list of words, each with a list of page numbers, an index can be implemented by chaining together each occurrence of a keyword. In Lace, each time the `\index` command is used, a button is placed on the page. This button allows the reader to go straight to the next page in the index chain, or to choose from one of those pages.

glossary Similar to to table reference, each occurrence of a keyword from a glossary has a button overlaid which, when pressed, causes the glossary definition to be displayed in a separate window.

3 A tour of Lace

How is Lace used? In what follows, the reader is referred to Fig. 1, which is a snapshot of Lace in use. The writer has created an 'ordinary' document, using generic markup and normal facilities for cross-referencing, indexing etc; this can be printed in the ordinary way, but can also go through a special TEX `dvi` to POSTSCRIPT converter which generates mouse-sensitive 'buttons' when displayed under NeWS (Lace can be seen as a simple TEX previewer), and splits the document into a series of 'leaves' which can be referenced separately or in sequence. We should stress that the 'leaves' enjoy the the high standards of TEX typesetting, whether maths, foreign characters or tabular work.

When we start to use Lace, a figure no longer appears by default, but if a reference to it is selected with the mouse, a separate window is opened for the picture. The following navigation methods are offered to the user (the pop-up walking menus on the left-hand side of Fig. 1):

1. A 'jump to page #' facility (the page number is selected from a menu)
2. Selecting next page or previous page
3. Returning to the page from which one arrived at the current page
4. Selecting a section of the document from the table of contents
5. Starting a new window by selecting an object from the list of figures or the list of tables (second menu on Fig. 1).

The navigation menu is available from any of the windows on the screen, permitting different directions to be explored in different areas of the screen. We may note that the use of POSTSCRIPT means that a very wide variety of software can be used to create pictures, and the text and pictures are fused with a single display medium, rather than an artificial distinction between text and bitmap or object-oriented graphics. Thus the plan in Fig. 1 was created using Adobe Illustrator, and the table hidden beneath it using LATEX's tabular mode. The typesetting nicety inherent in using software

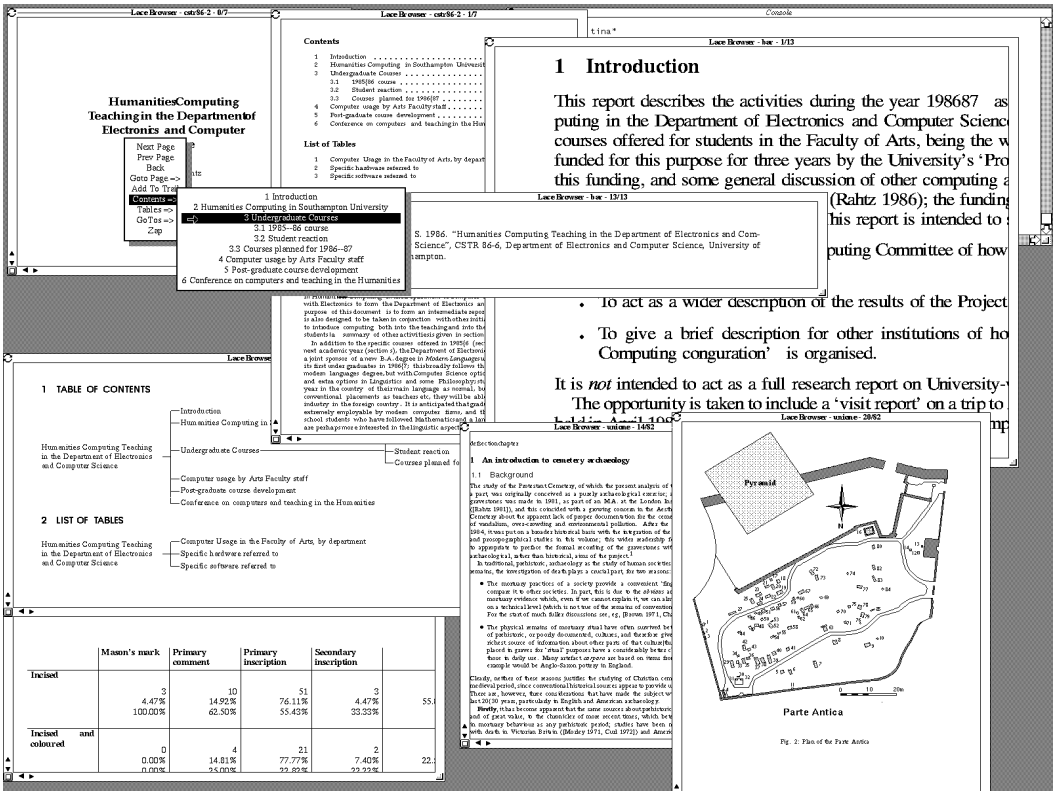

Figure 1: A Lace document in action

designed for producing books is noticeable; sections from several documents are shown, including Donald Knuth's example of his WEB(Knuth, 1983) system for documenting software.

The document elements are managed by the Lace server, a daemon process running on each participating machine. This means that any document may not only reference other elements of itself, but arbitrary elements of other documents (which will carry with them the normal apparatus of navigation menus). The process of requesting document elements can be extended arbitrarily to attach not only pictures or tables to points in the document, but also to access items in a picture archive, such as a videodisc. Fig. 1 shows the small videodisc controller window which is popped up by a click on references within the text; specifically the phrase 'Stone 66' in the bottom window is a button containing a request for a picture from a video database; the request has been satisfied, the picture is located and the controller is made available. In a similar way, traditional database accesses could be managed by the server, allowing one to bring up word definitions from a dictionary database (cf Raymond and Tompa, 1988, although the writers' electronic resources only stretch to the *Oxford Advanced Learner's Dictionary of Contemporary English*), or even to set off an SQL query to a relational system. We may imagine that, instead of saying 'see `\cite{formtable}`', in LaTeX parlance, we might write 'see `\lace{cemetery:table gravestones by form}`', which would remain valid markup, but which would provide a button asking the Lace server to contact a document called 'cemetery', which in fact consists of a series of SQL commands to create the table on the fly.

4 Conclusions

We referred above to Conklin's characteristics of hypertext; it is clear that Lace supports most of them (hierarchical strutures; non-hierarchical cross-reference links; arbitrary executable procedures; display graphics); the need for a graphical browser is provided by the table of contents, but some points remain which it is worth addressing here:

1. Hypertext systems sometimes permit paths to be strung together to make a single persistent object; this is only necessary, however, due to a lack of explicit authoring in the original preparation. In the Lace world, a new document can be created at the *source* level by a series of references to elements of other documents;
2. The versioning of nodes or links is done in the same way, by creation of new documents. This can be automated by requesting the Lace server to append a reference to the current object onto a path document.
3. Linear searching of documents for strings is tricky in Lace as it stands, due to the use of a 'compiled' version of the document for display. Although not currently instantiated, it will be possible in future to add the facility to search the source of a document, and return the correct section of the document. It must be stressed that Lace always deals in authored elements, not words, so that a search will not return the exact point in the document (the indexing mechanism is available for authors to flag appropriate words) but a jump to the area that the reader should be perusing.
4. It is a common complaint of browsing systems that they do not permit several people to edit it at once, or for readers to add comments. Lace proposes to address this issue with incrementally compiled annotations, which will be connected to the node.

Lace aims to minimize the effort needed to take a document to a new medium; the intention is to protect the writer's investment, and it must be stressed again that a document is seen here as an explicitly authored structure, rather than a simple 'dump' of facts at one level. We still see a place for explicit authoring, and 'hyperness' as an attribute which can be attached to a document; we also suggest that current hypertext does not distinguish sufficiently between the author and the writer , and the possible criticism of Lace, that it offers views on an essentially static system, may be regarded as a positive asset.

Bibliography

Akscyn, R. and McCracken, D. L. (1984). The zog approach to database management. In *Proceedings of the Trends and Applications Conference: Making Database Work*.

Akscyn, R., McCracken, D. L., and Yoder, E. (1988). Kms: A distributed hypertext system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–835.

Brown, P. J. (1986). A simple mechanism for authorship of dynamic documents. In van Vliet, J. C., editor, *Text Processing and Document Manipulation: Proceedings of the International Conference*, pages 34–42. Cambridge University Press.

Brown, P. J. (1987). Turning ideas into products: the guide system. In *Hypertext '87 Papers*.

Bryant, M. (1988). *SGML: an author's guide*. Addison Wesley.

Bush, V. (1945). As we may think. *Atlantic Monthly*, (176):101–108.

Conklin, J. (1987). Hypertext: a survey and introduction. *IEEE Computer*, (20):17–41.

Knuth, D. (1983). The WEB system of structured documentation. Technical report, Stanford.

Nelson, T. H. (1967). Getting it out of our system. In Schecter, G., editor, *Information Retrieval: A Critical View*, pages 191–210. Thompson Book Co., Washington, D.C.

Nelson, T. H. (1974). Dream machines: New freedoms through computer screens—a minority report. In *Computer Lib: You Can and Must Understand Computers Now*. Hugo's Book Service, Chicago, Illinois.

Raymond, D. R. and Tompa, F. W. (1988). Hypertext and the new oxford english dictionary. *Communications of the ACM*, 31(7):871–879.

Tebbutt, D. (1988). Guide. *Personal Computer World*.

Yankelovich, N., Haan, B., Meyrowitz, N., and Drucker, S. (1988). Intermedia: The concept and the construction of a seamless information environment. *IEEE Computer*, pages 81–96.