UNIVERSITY OF
NEWCASTLE UPON TYNE

University of Newcastle upon Tyne

COMPUTING SCIENCE

Models for Legal Documentation: Using Formal Methods for Quality Assurance in Hypertext Systems

B.N. Rossiter, T.J. Sillitoe and M.A. Heather

TECHNICAL REPORT SERIES

No. 464 December, 1993

TECHNICAL REPORT SERIES

No. 464 December, 1993

Models for Legal Documentation: Using Formal Methods for Quality Assurance in Hypertext Systems

B.N. Rossiter, T.J. Sillitoe and M.A. Heather

Abstract

Current hypertext systems have been widely and effectively used on relatively small data volumes. But the law is concerned with large complex data sets. The potential of database technology is explored for aiding the implementation of hypertext systems holding very large amounts of complex data. Databases meet many requirements of the hypermedium: persistent data management, large volumes, data modelling, multi-level architecture with abstractions and views, meta-data integrated with operational data, short-term transaction processing and high-level end-user languages for searching and updating data. To illustrate the potential for the use of data bases, a system implementing the storage, retrieval and recall of trails through hypertext comprising textual complex objects is described. Weaknesses in current database systems for handling the legal models are discussed.

Printed and published by the University of Newcastle upon Tyne,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

ROSSITER, Brian Nicholas

Models for Legal Documentation: Using Formal Methods for Quality Assurance in Hypertext Systems
[By] B.N. Rossiter, T.J. Sillitoe and M.A. Heather

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Science, 1993.

(University of Newcastle upon Tyne, Computing Science, Technical Report Series, no. 464)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.
Computing Science. Technical Report Series. 464
SILLITOE, Timothy John
HEATHER, Michael A.

Abstract

Current hypertext systems have been widely and effectively used on relatively small data volumes. But the law is concerned with large complex data sets. The potential of database technology is explored for aiding the implementation of hypertext systems holding very large amounts of complex data. Databases meet many requirements of the hypermedium: persistent data management, large volumes, data modelling, multi-level architecture with abstractions and views, meta-data integrated with operational data, short-term transaction processing and high-level end-user languages for searching and updating data. To illustrate the potential for the use of data bases, a system implementing the storage, retrieval and recall of trails through hypertext comprising textual complex objects is described. Weaknesses in current database systems for handling the legal models are discussed.

About the author

Nick Rossiter is a Lecturer in the Department of Computing Science at the University of Newcastle upon Tyne with particular interests in databases and system analysis.

Tim Sillitoe was an M.Sc. student in the Department of Computing Science at the University of Newcastle upon Tyne from 1988-89.

Michael Heather is Senior Lecturer in Law where he has been responsible for computers and law since 1979.

Suggested keywords

DATABASES	FORMAL METHODS	HYPERTEXT	LEGAL DOCUMENTS
MODELLING	QUALITY CONTROL	PETRI NETS	SEMANTIC MODELS

Suggested classmarks (primary classmark underlined)

Dewey (18th):	001.6442	029.7	348.022
U.D.C.	681.322.06	651.838.8	340.13

Models for Legal Documentation: Using Formal Methods for Quality Assurance in Hypertext Systems

B.N.Rossiter & T.J.Sillitoe
Computing Science
Newcastle University, NE1 7RU

M.A.Heather
Department of Law
University of Northumbria at Newcastle, NE1 8ST, UK

November 1993

Abstract

Current hypertext systems have been widely and effectively used on relatively small data volumes. But the law is concerned with large complex data sets. The potential of database technology is explored for aiding the implementation of hypertext systems holding very large amounts of complex data. Databases meet many requirements of the hypermedium: persistent data management, large volumes, data modelling, multi-level architecture with abstractions and views, meta-data integrated with operational data, short-term transaction processing and high-level end-user languages for searching and updating data. To illustrate the potential for the use of data bases, a system implementing the storage, retrieval and recall of trails through hypertext comprising textual complex objects is described. Weaknesses in current database systems for handling the legal models are discussed.

About the author

Nick Rossiter is lecturer in the Department of Computing Science with particular interests in databases and systems analysis.

Tim Sillitoe was an M.Sc. student in the Department of Computing Science from 1988–89.

Michael Heather is senior lecturer in law where he has been responsible for computers and law since 1979.

Suggested Keywords

Modelling, legal documents, semantic models, Petri Nets, formal methods, quality control, hypertext, databases.

Chapter 1

Introduction

Legal information is large in terms of data volumes and complex in structure both within documents and in terms of links between different documents. Such a combination of bulk and complexity suggests the use of databases which need to be distinguished from databanks [Rossiter 1985]. Legal databanks have been increasing their capability in an ad hoc manner with the result that it is often not possible to connect them together or use more advanced hypertext techniques on them. For this, there has to be some universal structure recognition. This is achieved by formal methods. Formal methods in programming software tends to mean the use of languages like Z and VDM. In databases, the formalism is at a higher level and the whole subject of databases results in a consistent model being developed. Languages tend to focus on the data content of databases but a database exists without any data. It is the structure which is developed for the particular legal data that is to be held. Complex heterogeneous databases operate in a hypermedium.

The hypermedium is an information space representing a high level abstraction of data. It represents an idealised view of the information needs of an area of particular human interest or activity. Information usually amounts to connections between different items to be found in human experience. These may be physical things or they may be ideas.

The significant feature of the hypermedium is the nature of this connection between data. It consists of an ordering but an ordering that is not unique. Many possible orderings may exist. While the computer is an obvious tool for handling and organizing large quantities of data in the hypermedium, straight-forward procedural methods cannot cope with the complexity of the organization. The experience of early workers in databases is being repeated in the hypermedium by those engaged in developing hypertext. To progress beyond small simple systems requires the writing of what amounts to a customized database system. However, in adopting a customized solution, there is an immediate loss of generality and of functionality and a deterioration in quality. The hyperbases so developed may only be usable in their home environment whereas a generalised database implementation would provide the basis for the use of the same information for many other purposes [Rossiter & Heather 1990]. There is the matching and retrieval capabilities of information retrieval systems, the document segmentation and word indexing of free text products, the display of mark-up languages, the layouts and layers to be found in the Office Document Architecture, the use of metadata for data exchange, and the application of a body of rules as in the field of AI and expert systems.

It therefore seems better to make use of the experience of the database community in building large hyperbases but it cannot be pretended that the benefits of one technology to the other are all in one direction. As will be seen later, database technology in its present form has some deficiencies in modelling complex objects and events, the solution of which will be given greater impetus by involvement in new challenging areas. The authors therefore see the relationship between database and hypertext technologies as symbiotic rather than parasitical. The hope is that database technology is both extending to the hypermedium and being extended by it.

For reasons of continuity from the old, a fundamental unit of data in the new hypermedium is a document. Present hypertext provides mainly for small simply structured documents and in the way that it concentrates on factors at the human-machine interface, it gives good insight into the capabilities needed for a full hypermedium system. Three main types of link are recognized in hypertext systems:

1. explicit inter-document links representing citations,
2. lexical links in which the meaning of words is resolved,
3. conceptual links in which implicit semantic connections are made between one document and another.

The work described later is mostly concerned with symbolic links between one document and another. Lexical links pose greater difficulties in implementation because of frequent ambiguity in finding the definition of a word amongst its many usages in a text. Implicit links have proved to be difficult for the machine to locate automatically but can be entered manually by the user in most hypertext systems and in small-scale applications can provide very rich structures. It is unlikely that such richness can be achieved in large hyperbases where automated authoring is likely to prevail.

In traditional document systems, there is often a very arbitrary division in information [Rossiter & Heather 1989] because of the rigidity enforced by predefined document sizes. In hypertext systems, this is overcome to some extent through various composition techniques for representing 'isPartOf' relationships. Through such aggregation, logical documents can be defined which are a synthesis of what may be many diverse physical documents. The view of the authors is that these ideas need developing further to represent a document as a complex data object holding information in the form of structured data. The representation of document structures in database models is investigated more fully later.

1.1 Limitations of Current Hypertext Systems

Present hypertext systems concentrate on the human-computer interface and rely on semi-automated or manual techniques to represent links between one document and another. This is satisfactory for small simple document structures but otherwise there are a number of problems:

- The use of symbolic addressing is not fully exploited to cope with pre-existing forms of citation and for automated authoring of large quantities of text.
- Methods of management of persistent data are relatively primitive.
- Searching facilities are specialised.
- There is no consensus on the nature of the formal data model which is necessary to provide an integrated framework for data structuring and manipulation. Recent work employing set theory [Tompa 1989],[Garg 1988], Petri Nets [Furuta & Stotts 1989] and Z [Hitchcock & Wang 1992] shows the urgency with which this area is now being tackled. It is important for large complex applications that current hypertext practice involving the use of directed graph (general network) structures, inheritance hierarchies and object-oriented scripts be underpinned by a greater body of theory. A formal storage model using network structures has been developed [Campbell & Goodman 1988] but this omits many of the activities.
- Node data is WYSIWYG. There is limited opportunity for mapping and indirection between user views and storage structures. There tends to be

one fixed view - that of the author, with little scope for the preferences of individual readers.

- Hypertext systems are generally self-contained and cannot be easily integrated with other programs and data. It is difficult for another application to use the hyperbase.

These problems are emphasised with large data volumes, multiple authorship, complex inter-node and intra-node relationships, need for multiple views of same hypermedium, and a desire to integrate the hyperbase with other types of application within the organization.

1.2 Potential of Database Technology

Database technology has significance as it can assist in many of these problem areas: high-level end-user languages such as SQL can be embedded in standard programming languages to integrate database facilities with other functional aspects; management of large volumes of persistent data, including such aspects as security, integrity, concurrency and optimization of access, is a central tenet of the technology; multi-level architectures with mappings from logical to physical levels provide different views of the same stored data; content-addressing can be integrated with navigation to give facilities as sophisticated as those found in information retrieval systems.

The use of data models requires more detailed discussion. Database technology depends on the development of an appropriate data model for structuring and manipulating the data. It could be argued that the use of any model is reductionist, resulting in a loss of information. However, a data model does provide a rigorous framework within which an application can be developed. It therefore seems necessary, to exploit the full power of hypertext, to have some machine model expressing semantic detail of the documents held with a full abstract specification of the data-types involved and a multi-level architecture similar to that of a DBMS. A clear problem is the kind of model that is most suitable for representing the architecture of documents and multimedia data and for providing usable query languages. As will be seen later, current DBMS models are inadequate in some respects.

The manner in which cross-references are realised and checked is crucial for a consistent hyperbase. Database systems employing symbolic keys for identification of objects have an inherent advantage over less conceptual approaches in handling text whose content is continuously changing. In first generation hypertext systems with physical node addressing, cross-references must in advance be fully identified as in a network database. In a value-oriented database approach to hypertext, links are made dynamically at run-time using symbolic key matching techniques. Both means provide for display and navigation through documents. The physically-oriented approach uses less resources but the early binding of identifier to data is more of a static method which allows less flexibility if, for example, data is being loaded in an uncertain order or key values are being changed or deleted.

The greater flexibility obtained through the dynamic power of lazy evaluation using database technology is not the only advantage in this area. Constraints like referential integrity can be placed automatically on new data entered into the system and on updates to existing records. Potential cross-references in symbolic form are checked against the current database and must succeed for the new or changed record to be accepted without reservation. At the programmer's discretion, errors resulting from dangling cross-references can result either in the new data being rejected or accepted with reservation. Such reservations include a warning message, flagging of the citing field or a setting of the citing field to null.

The various levels of verification of links makes the construction of large hyperbases a very much easier and rigorous process through a multi-stage commit process. During the addition of user data, dangling cross-references, perhaps reflecting the order in which data is added, are flagged in the first pass and only after a second pass to re-check citations is the possibility of rejection considered. In any event, cross-references which cannot be resolved will remain flagged as such so that the system is always consistent with respect to which references are navigable. Finally the concept of referential transparency should be raised. In a database environment, the entire management of the links will be automatically handled by the system to relieve the user of all responsibility for maintaining referential integrity.

Chapter 2

An Example Document Architecture

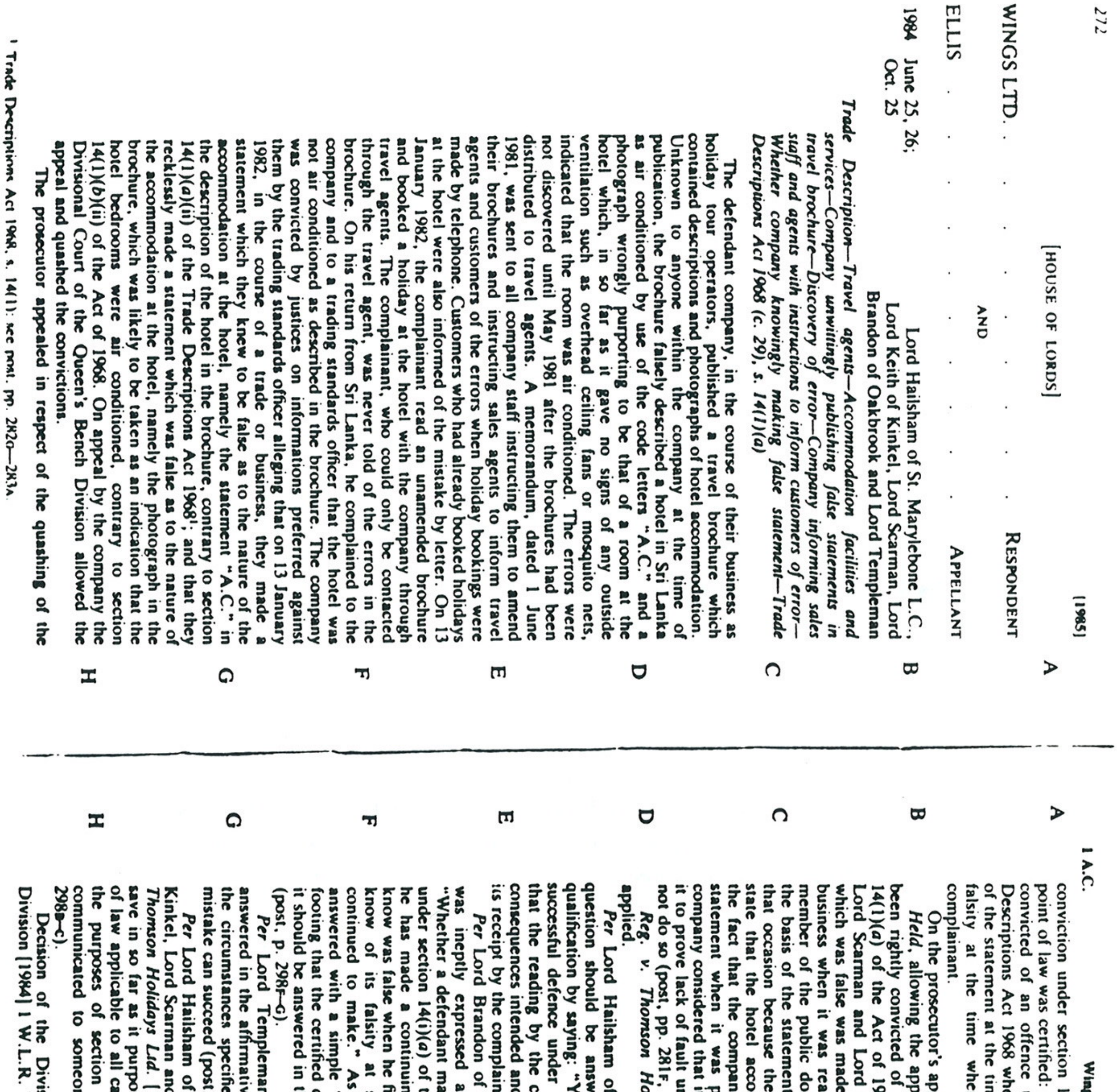
In order to examine document architectures, the example of English legal statutes will be used in this paper. In England, Parliament enacts statutes and Figure 2.1 shows an extract from a case to illustrate the complex relationships between this text and statute documents. There are a number of references from the case to the Trade Descriptions Act and also to other cases and to different parts of the same case.

Within statutes, a section represents the smallest self-contained free standing unit of text although subsections may be directly cited sometimes. A section is a mere point in the textual hypermedium and can rarely be consulted alone or understood without reference to other documents. For many purposes, sections are grouped together into parts or paragraphs into schedules. As any of the information in the Figure 2.1 may have a bearing on a section in question, it can readily be seen that advanced hypertext features are needed if all the relevant subject matter is to be available and easily reached in the electronic medium. Our work can be contrasted with that of Yoder & Wettach [1989] who have also developed a hypertext system for the law. Their system is very flexible in the forms of data accepted but lacks a formal data model for controlling structures and for providing a general means of manipulating the data.

2.1 Trails and Paths

The existence of conceptual paths through textual documents was recognised by Bush in 1945 [Bush 1945]. Treu [1971] considered the existence of trails through bibliographic citations and thought they should be preserved for a searcher to retrace his steps at a later date. At Newcastle, the need to provide a conceptual framework for the machine to assist the human in his database searching and navigation was recognised in 1987 [Rossiter 1987] with a prototype implementation of the recording of trails in database tables as persistent data fully integrated with the hypertext data. The main objective of the trails was to assist the human in communication with the machine by removing the need to memorise backward and forward references, unsuccessful routes through the database, search terms used and the search and navigation strategy. Also in 1987, Conklin [1987] identified one of the major difficulties in current hypertext systems as the user becoming "lost in hyperspace" as a result of losing his way along a trail as a result of the demands made during navigation. Zellweger [1989] has classified the various kinds of path and emphasised the importance of implementing paths as first class data. Although the implementation of the paths as scripts is satisfactory for single-user systems, there are problems with sharing of the path data in multi-user environments.

There is general agreement in the work quoted above that path information should be first-class data, replayable with or without variation and an essential part of the user interface. Before considering the required structures in more detail, we will first consider database models for representing the internal structure of the statute.



272 [1985]

A [HOUSE OF LORDS]

WINGS LTD. . . . . RESPONDENT

AND

ELLIS . . . . APPELLANT

B 1984 June 25, 26; Oct. 25 — Lord Hailsham of St. Marylebone L.C., Lord Keith of Kinkel, Lord Scarman, Lord Brandon of Oakbrook and Lord Templeman

Trade Description—Travel agents—Accommodation facilities and
services—Company unwittingly publishing false statements in
travel brochure—Discovery of error—Company informing sales
staff and agents with instructions to inform customers of error—
C *Whether company knowingly making false statement—Trade*
Descriptions Act 1968 (c. 29), s. 14(1)(a)

The defendant company, in the course of their business as
holiday tour operators, published a travel brochure which
contained descriptions and photographs of hotel accommodation.
Unknown to anyone within the company at the time of
publication, the brochure falsely described a hotel in Sri Lanka
D as air conditioned by use of the code letters "A.C." and a
photograph wrongly purporting to be that of a room at the
hotel which, in so far as it gave no signs of any outside
ventilation such as overhead ceiling fans or mosquito nets,
indicated that the room was air conditioned. The errors were
not discovered until May 1981 after the brochures had been
distributed to travel agents. A memorandum, dated 1 June
1981, was sent to all company staff instructing them to amend
E their brochures and instructing sales agents to inform travel
agents and customers of the errors when holiday bookings were
made by telephone. Customers who had already booked holidays
at the hotel were also informed of the mistake by letter. On 13
January 1982, the complainant read an unamended brochure
and booked a holiday at the hotel with the company through
travel agents. The complainant, who could only be contacted
through the travel agent, was never told of the errors in the
F brochure. On his return from Sri Lanka, he complained to the
company and to a trading standards officer that the hotel was
not air conditioned as described in the brochure. The company
was convicted by justices on informations preferred against
them by the trading standards officer alleging that on 13 January
1982, in the course of a trade or business, they made a
statement which they knew to be false as to the nature of the
G accommodation at the hotel, namely the statement "A.C." in
the description of the hotel in the brochure, contrary to section
14(1)(*a*)(ii) of the Trade Descriptions Act 1968[1]; and that they
recklessly made a statement which was false as to the nature of
the accommodation at the hotel, namely the photograph in the
brochure, which was likely to be taken as an indication that the
hotel bedrooms were air conditioned, contrary to section
H 14(1)(*b*)(ii) of the Act of 1968. On appeal by the company the
Divisional Court of the Queen's Bench Division allowed the
appeal and quashed the convictions.
The prosecutor appealed in respect of the quashing of the

[1] Trade Descriptions Act 1968, s. 14(1): see post, pp. 282D—283A.

1 A.C. Wings Ltd. v. Ellis (H.L.(E.))

A conviction under section 14(1)(*a*) of the Act and the following
point of law was certified: whether a defendant may properly be
convicted of an offence under section 14(1)(*a*) of the Trade
Descriptions Act 1968 when he has no knowledge of the falsity
of the statement at the time of the publication but knew of the
falsity at the time when the statement was read by the
complainant.
On the prosecutor's appeal:—
B *Held*, allowing the appeal, that the defendant company had
been rightly convicted of committing an offence under section
14(1)(*a*) of the Act of 1968 since (*per* Lord Keith of Kinkel,
Lord Scarman and Lord Brandon of Oakbrook) a statement
which was false was made by the company in the course of its
business when it was read by the complainant, an interested
member of the public doing business with the company upon
the basis of the statement; that the offence was committed on
C that occasion because the company then knew it was false to
state that the hotel accommodation was air conditioned and
the fact that the company was unaware of the falsity of the
statement when it was published was irrelevant; that if the
company considered that it was innocent of fault, it was open to
it to prove lack of fault under the statutory defences, but it did
not do so (post, pp. 281F, 290E, 297C–D, 298F–G).
D *Reg.* v. *Thomson Holidays Ltd.* [1974] Q.B. 592, C.A.
applied.
Per Lord Hailsham of St. Marylebone L.C. The certified
question should be answered in the affirmative but with a
qualification by saying: "Yes, unless the defendant has raised a
successful defence under section 24 of the Act and provided
that the reading by the complainant was part of the chain of
consequences intended and authorised by the defendant prior to
E its receipt by the complainant" (post, p. 290D–E).
Per Lord Brandon of Oakbrook. (i) The certified question
was ineptly expressed and should be amended as follows:
"Whether a defendant may properly be convicted of an offence
under section 14(i)(*a*) of the Trade Descriptions Act 1968 when
he has made a continuing false statement, which he did not
know was false when he first made it, but which having come to
F know of its falsity at some later time, he has thereafter
continued to make." As amended in that way, it should be
answered with a simple "Yes." (post, p. 298E–F). (ii) On the
footing that the certified question can be answered as it stands,
it should be answered in the manner proposed by Lord Scarman
(post, p. 298F–G).
Per Lord Templeman. The certified question should be
G answered in the affirmative. The accused must plead and prove
the circumstances specified in section 24 before a defence of
mistake can succeed (post, pp. 300G, 301A).
Per Lord Hailsham of St. Marylebone L.C., Lord Keith of
Kinkel, Lord Scarman and Lord Brandon of Oakbrook. *Reg.* v.
Thomson Holidays Ltd. [1974] Q.B. 592 was correctly decided
save in so far as it purports to decide as a general proposition
of law applicable to all cases that a statement is only made for
H the purposes of section 14 of the Act of 1968 when it is
communicated to someone (post, pp. 285D–F, 290E, 296D–E,
298B–C).
Decision of the Divisional Court of the Queen's Bench
Division [1984] 1 W.L.R. 731; [1984] 1 All E.R. 1046 reversed.

Figure 2.1: An example of a Case showing Statute Citations

Chapter 3

Database Models and Textual Structures

The basic DBMS models such as the relational are not suitable for manipulation of the fine structure of documents mainly due to the problems of normalization and aggregation of textual data [Rossiter & Heather 1990] which in general terms result from an inadequate representation of complex objects. At least for representing ideas, it is necessary to move on from the classical models to the semantic models because the required emphasis is on capability, expressiveness and abstraction. A range of semantic models incorporating more features and constraints than in the basic models has been proposed in an attempt to model more closely the real world. These include the Entity-Relationship (E-R) Model [Chen 1976] and Taxis [Mylopoulos et al 1980], both of which have been employed in this work.

3.1 Class Structures

A Chen E-R diagram of English statutes is shown in Figure 3.1. More details on this model, which is based on directed–graph theory, have been presented elsewhere [Rossiter & Heather 1990]. Two types of hierarchy are embedded within the class structure:

- An essential inheritance hierarchy to indicate the inheritance of properties (attributes) automatically by lower level objects from higher ones through 'isA' relationships.

- An aggregation hierarchy to indicate potential groupings of data through 'is-PartOf' relationships. This hierarchy provides the framework upon which textual units are dynamically aggregated to satisfy varying user requirements.

The aggregation hierarchy has as its root a highly abstract object *node* which has some similarity to a node in hypertext terminology comprising a chunk of data for presentation to the user. There are thus clear similarities between the two approaches. However, there are important differences:

- in hypertext systems, nodes are static structures at run-time whereas in our approach, a node can be dynamically generated at any time from any of the underlying text objects by aggregation.
- in hypertext systems, the internal structure of the nodes can be left undefined whereas in database technology there is a clearly defined structure for each specific text object at lower levels of the class hierarchy.
- the aggregation of *node* in our approach is always made in the context of symbolic identifiers (see Figure 3.2) rather than record or card numbers.

3.2 Symbolic Addressing for Hypertext

For navigation in the hypermedium, it is important to be able to identify uniquely individual units of text so that cross-references can be resolved. With the complex object structure employed in this study, it has been found that the optimal solution is to employ a generic symbolic key *all.unit.id* for the abstraction *node* as shown in Taxis-like form in Figure 3.2. The key *all.unit.id* effectively defines a generic heading which contains an integer value for each possible component of a textual identifier. The form of the key is application dependent: in our work, nine different components have been identified such as section, subsection and footnote. For a given instance of a text, the values of some components are inapplicable. Such components have a value of zero: all other components have positive values, for example, *section#* would be assigned the value 6 in the heading of the sixth section of an act. This provides a completely general mechanism for addressing all objects in the inheritance hierarchy. The values for the attributes of *node* are constrained by the variables such as *ssmin* and *ssmax* which specify the minimum and maximum values permitted for subsection numbers.

The class *text* is a specialization of *node* representing an abstraction of the main body of text. As shown in the definition of *text.id*, a subset of the components of the generic key *all.unit.id* is required to address the main text. Specific features included in *text* but not in *node* are attributes representing various details of the internal structure of an item of text. Cross-references are represented by the class *XRef* with each citation held in *ref.id* comprising a pair of symbolic identifiers for the citing and cited text units, respectively. The constraint is specified that the citing and cited objects must be members of the set *text*: therefore, the identifiers of the text units must conform to the structure of *text.id* and the text units must be instances of the class *text* to enforce referential integrity.

3.3 Models for Expressing Dynamic Aspects of Trail Management

Life–cycles are represented well in Petri Nets [Reisig 1985]. Their significance for representing dynamics of law was recognised very early on [Holt & Meldman 1971]. At this time of the 25th anniversary of the Italian National Research Council's Institute of Legal Documentation, it is appropriate to recall the pioneering work in this field, with which the Institute was connected, by Enrico Maretti [Degli Antoni & Zonta 1982]. Figure 3.3 shows an Entity–Life History Model in Petri Net form for the entity–type *path* which indicates the sequence and choice of events in searching and navigating and in recording and replaying trails through the hypermedium. The diagram shows the processes involved (in rectangles), the states reached between processes (in circles) and the order of execution of the processes (through the direction of the arrows). Selection is represented by multiple outputs from a state, conjunction by multiple inputs into a process, iteration by cycles in the flow of processing and parallelism by multiple outputs from a process. Whilst execution of a particular process is not complicated, it is a matter of integrated management of the very large number of processes that are possible and their complex inter-relationships. It is interesting to note that Petri Nets have also been employed by Furuta & Stotts [1989] for representing the semantics of dynamic activities with documents. The formal network basis for the representations is an attractive feature.

Three types of information are accessed by the processes shown in Figure 3.3: the hypermedium itself, the names of the trails made by each user held in *path* and a complete history held in *pathitem* of each path comprising an initial content-based search followed by a series of navigational commands. The structure of this information is described in the next section.

3.4 Models for Expressing Static Aspects of Trail Management

As companion to the Petri Net of Figure 3.3, there is an E-R diagram in Figure 3.4 to show the relationships between the entities holding the trail information *path* and *pathitem* and other entities relevant to trail management. Each user can hold many paths each of which holds many path items. For branching trails [Zellweger 1989], it is necessary to introduce the involuted relationship *cites* to indicate that a single path item can branch to many other path items during navigation through the user backtracking. For linear trails, the relationship *cites* is not required.

The entity-type *Current.Record.Position* has been introduced to explicitly indicate the current selected object. Many users can be active at a given time but it is an assumption at present that each user holds a single current record position at any given time. The entity-type *hypermedium* is in a 1:N relationship with *pathitem* indicating that each hypermedium object can appear many times as a path item but that each path item refers to only one hypermedium object. The importance of the relationship *item.found.in* for integrity of the trail is described later.

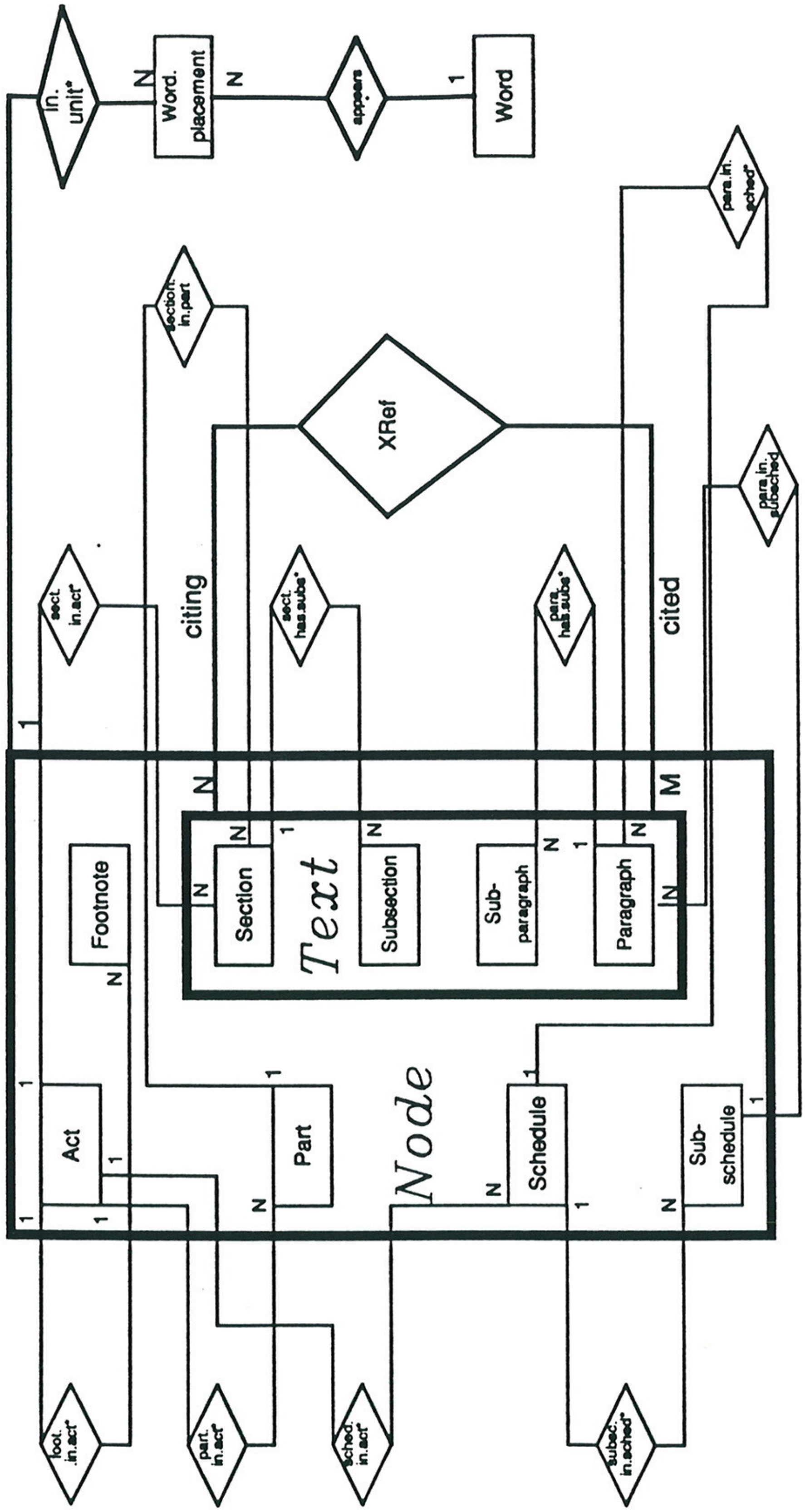


Figure 3.1: The Chen E–R Diagram for Statutes

```
define      AnyDataClass Node with
            ss#: {| ssmin:ssmax |}
            section#: {| sectmin:sectmax |}
            part#: {| partmin:partmax |}
            subp#: {| subpmin:subpmax |}
            para#: {| paramin:paramax |}
            subschedule#: {| subsmin:subsmax |}
            schedule#: {| schmin:schmax |}
            footnote#: {| footmin:footmax |}
            year: {| yearmin:yearmax |}
            chapter: {| chapmin:chapmax |}
unique
            all.unit.id: (year, chapter, part#,
            section#, ss#, schedule#, subschedule#,
            para#, subp#, footnote#)

define      AnyDataClass Text isA Node with
changeable
            marginal.note.other: string
            crossnotes: string
            omissions: string
            footnotes.old.stats: string
            formatting.attribute1: string
            formatting.attribute2: string ... etc
unique
            text.id: (year, chapter,
            section#, ss#, schedule#, para#, subp#)

define      AnyDataClass XRef with
            citing.text.id: set of Text
            cited.text.id: set of Text
unique
            ref.id: (citing.text.id, cited.text.id)
```

Figure 3.2: Taxis-like Specification of Symbolic Key for Statutes

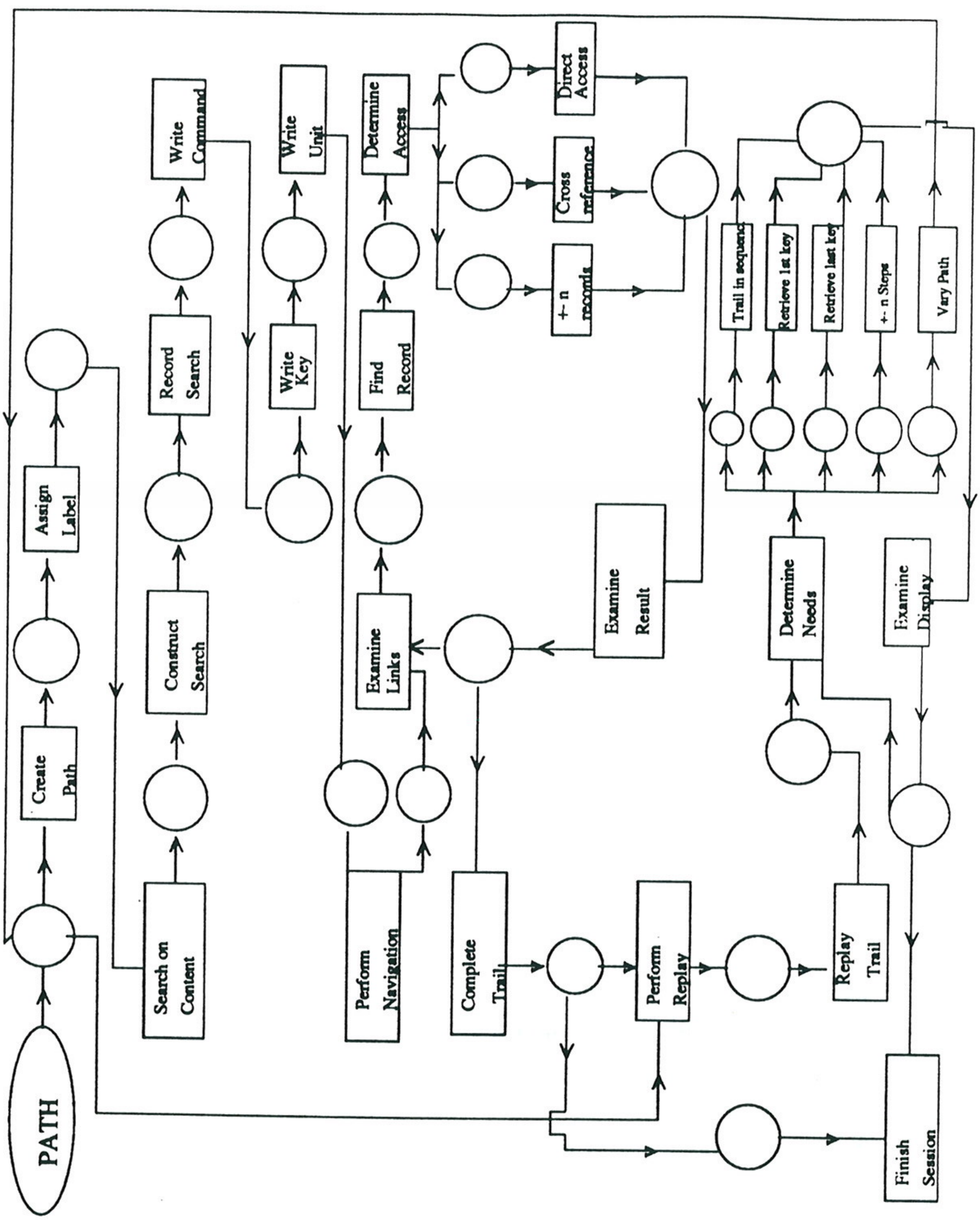
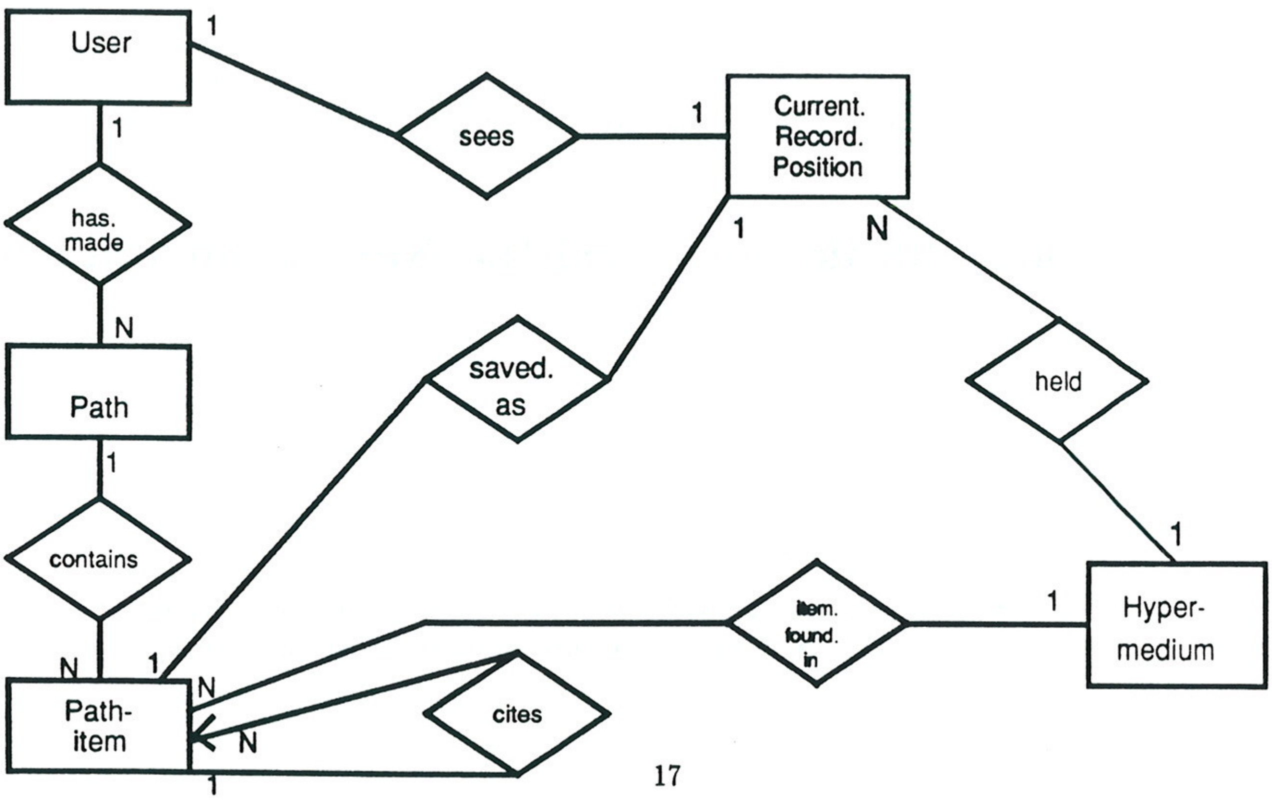


Figure 3.3: Entity–life History Model for Trail Management in Petri Net Form

Figure 3.4: E-R Diagram for Static Aspects of Trail Management



Chapter 4

Implementation of Trail Management System

The system was implemented on the SPIRES DBMS from Stanford University run on an Amdahl 5860 of the NUMAC service. The textbase STATLT holding the statutes for England has been developed and refined in a series of projects since 1980 and at the start of the project described here already provided a very detailed definition of the data structure, full text searching facilities, symbolic addressing in the manner of Figure 3.2 and a multivalued attribute *marg-note-xref* in each text unit to record cross-references made to other parts of the text [Rossiter & Heather 1987].

The current work is concerned with the implementation of the dynamic aspects shown in Figure 3.3 and the static aspects of Figure 3.4. The additional tables created to record the status of navigation will first be described.

4.1 Tables to record the Navigational Status

The entity-types *path* and *pathitem* shown in Figure 3.4 hold all information on the trails made by users through the textbase. The attributes describing this information are shown below (key attributes in bold):

- path(**user.id, trail.num,** trail.label)
- pathitem(**user.id, trail.num, command.num,** command, citing.text.id, cited.text.id, current.unit, link.status, relevance)

Each trail is labelled with a string *trail.label* for identification by the user. In *pathitem*, *cited.text.id* holds the symbolic key of the current record after the command held in *command* has been both executed and successful. Success or failure is indicated by the value for the logical attribute *link.status*. The current unit size, indicating the extent to which the complex object structure has been aggregated to provide results to the user, is indicated by the value for *current.unit*. The attribute *relevance* can be used to record the desirability of taking a particular route.

The attribute *citing.text.id* represents the involuted relationship *cites* of Figure 3.4 and is used as a backward reference point to enable the user to perform backwards and forwards tracking through the text. The attribute pair *citing.text.id* and *cited.text.id* is exactly equivalent to *ref.id* defined earlier in the Taxis-like symbolic key definition of Figure 3.2. The tables and their attributes are extensively used by the processes described in the Petri Net of Figure 3.3.

4.2 Dynamic Aspects and the User Interface

The processes shown in Figure 3.3 were implemented using the SPIRES Protocols language. Two types of command are recognised by the system. SPIRES system commands are passed to the database kernel without modification. Other commands to validate and execute either a search, navigation or trail request are parsed and then sent to the appropriate process. It should be emphasised that the interpretation of users' actions is to some extent context-driven. Thus if the variable *status* holds the value REPLAY, the users' actions will be interpreted as far as possible as involving the recall of a trail. If the value is ACTIVE, the user is thought to be navigating and if INACTIVE (from the navigation perspective) performing an initial search to locate a record on content prior to navigation. However, if it is unambiguous that a user wishes to change his mode of operation from, say, navigation to content search, his status will be changed transparently from, in this case, REPLAY to INACTIVE. This flexibility is very important as it is only by changing mode in the middle of a session that a user can vary an earlier trail to explore the text in a new manner. The facilities available to the user under each status value are as follows:

- INACTIVE: A search command creates an initial result stack of items. This is followed by iterative searching with Boolean logic on the current stack. Navigation can only sensibly proceed when the user has identified a single record as of initial importance from content searching. The ideal is probably an initial list of ranked records as described by Croft & Turtle [1989].

- ACTIVE: Navigation commands available are of three main types:
- entering a positive or negative number enables the user to browse backwards or forwards through the text in logical sequence of the textual units. This command is typically used for browsing in either direction through sections within a part or paragraphs within a schedule at a constant textual unit size.
- entering the command **ref** directs the system to find the record referenced by the current record. If several records are referenced from a single record, the user will be given a choice as to which one is required. If the reference is to a high-level unit such as a *part*, objects will be aggregated to retrieve a complete *part* for the user. This command can therefore dynamically change the current textual unit size.
- entering values for the identifiers of subobjects of the symbolic key defined in Figure 3.2 finds the record with symbolic key with new values for the designated subobjects and current values for other components. The current textual unit size is adjusted accordingly.

With all three forms of the navigation command, execution results in updating the table *pathitem* defined earlier and, if successful, making the object found the current item.

- REPLAY: for the replay of trails established earlier, the user first provides a string *trail string* for identifying the required trail held in the table *path*. If the trail exists, the first action held for the path in *pathitem* will be executed and the system status will be changed to REPLAY. During the replay of a trail, a user can enter any of the following:
- *first* finds the key of the record found at the beginning of the selected trail and establishes it as the current record.
- *last* finds the key of the record found by the end of the selected trail and establishes it as the current record.
- *fwd[n]* takes the navigation n steps forward from the current position.
- *bwd[n]* returns the navigation back n steps.
- *end* causes the status of the system to be changed from REPLAY to INACTIVE.

Chapter 5

Discussion

We have used current database techniques to satisfy our requirements. Of particular interest is the availability of both powerful browsing and searching facilities, the recording of all information concerning user trails as persistent data in fully-fledged database tables, and the dynamic variation of text unit size to meet changing user demands.

However, our task was relatively hard in two areas:

1. the dynamic adjustment of unit size; and
2. the integration of dynamic and static models.

In our implementation, aggregation was achieved at run-time through masking out components of the primary key and assembling, using the Protocols language, the series of text objects meeting the criteria implied by the user's current request. Reasonable performance was achieved in this task but the aggregation is being achieved by external operations on the objects rather than by the more conceptual approach of aggregation abstraction: new object classes with aggregation methods are defined to represent the various unit sizes.

Database technology does not provide a completely satisfactory solution to this problem. The definition of abstract data types as in the well–known commercial ISO–standard relational database system Ingres to represent the various aggregation possibilities may give problems with closure: the return of a multi–valued set produces an unnormalized relation. Alternatively, an object–oriented database system such as GemStone could have been employed. This would have modelled well the inheritance abstractions but aggregation is achieved by external operations on objects as in our current implementation.

The dynamic and static aspects have been implemented using different models which are weakly-integrated. This lack of integration is found in all conventional database systems in current use [Tsichritzis & Nierstrasz 1988]. On the other hand, an inherent feature of object-oriented databases is that methods do form part of the class definition. Some semantic database models such as Taxis also provide this capability and their expressiveness has been examined for text [Rossiter & Heather 1990]. Although these integrated models are currently at the experimental stage for realistic amounts of data, their employment in future large hyperbase systems seems almost obligatory. The object-oriented model of hypertext developed using the Vienna Development Method [Lange 1990] shows the potential of the paradigm in this area.

In addition, there is also a number of areas where further work is required:

1. The interface provided to users. Layered object-oriented techniques employing multi-windowing need to be front-ended onto the present system.

2. Investigation of the semantics of trail integrity. The integrity of trails depends during their existence on no component object being deleted during maintenance of the hypermedia database. There is therefore a need for restrictions on the actions that are permitted on objects that participate in trails. Operations such as deletion on any *hypermedium* object participating in the relationship *item.found.in* should perhaps be constrained. Further work is needed at the conceptual level in this area to determine the exact nature of the constraints required.

Chapter 6

Conclusions

Hypermedia systems are very complex: events have to be controlled over long periods, as in the design, control, maintenance and integrity of linear and branching trails used for navigation; text and graphical information comprises complex data objects with the need for aggregation and inheritance abstractions; and interfaces must employ multi-windowing techniques and be natural according to psychological models. We have considered hyperbases in terms of formal models based on Petri Nets for activity and directed graphs for data structures. However, there is some difficulty in integrating the two models to provide a unified formalism. We are currently investigating the use of category theory for providing a more powerful universal formalism for advanced computer applications [Rossiter & Heather 1993].

Chapter 7

References

V.Bush (July 1945), As we may think, Atlantic Monthly 101–108. Reprinted 1988 in: Computer-Supported Cooperative Work: A Book of Readings, ed. I. Greif, Morgan Kaufman 17–34.

B.Campbell and J.M.Goodman (1988), HAM: A General Purpose Hypertext Abstract Machine CACM **31** 855–861.

P.P–S Chen (1976), The Entity-Relationship Model - towards a unified view of data, ACM Transactions on Database Systems 1(1) 9–36.

J. Conklin (1987), Hypertext : An Introduction and Survey, IEEE Computer **20**(9) 17–41.

W.B.Croft and H.Turtle (1989), A Retrieval Model for Incorporating Hypertext Links, in: Hypertext'89 Proceedings; Special Issue - SIGCHI Bulletin 213–224.

G.Degli Antoni & B.Zonta (1982), Analysis of Laws by Petri Nets: Motivations and Methodologies, in: Artificial Intelligence and Legal Information Systems, C.Ciampi (ed), North–Holland **1** 273–299.

R.Furuta and P.D.Stotts (1989), Programmable Browsing Semantics in Trellis, in: Hypertext'89 Proceedings, Special Issue - SIGCHI Bulletin 27–42.

P.K.Garg (1988), Abstraction Mechanisms in Hypertext CACM **31** 862–870.

M.A.Heather and B.N.Rossiter (1987), Database techniques for text modelling: the document architecture of British statutes, University of Newcastle upon Tyne, Computing Laboratory Technical Report no 227.

M.A.Heather and B.N.Rossiter (1989), A Generalized Database Management Approach to Textual Analysis, in: Proceedings. 2nd International Colloquium, Bible and Computer: Methods, Tools, Results, Champion-Slatkine, Paris-Geneva 519–535.

P.Hitchcock & B.Wang (1992), Formal Approach to Hypertext System based on Object–oriented Database System, Information & Software Technology **34**(9) 573–592.

A.W.Holt & J.A.Meldman (1971), Petri Nets and Legal Systems, in: Jurimetrics Journal **12** 2.

D.B.Lange (1990), A Formal Model of Hypertext, in: Proceedings Hypertext Standardization Workshop, edd. J. Moline, D. Benigni and J. Baronas, National Institute of Standards and Technology 145–166.

J.Mylopoulos, P.A.Bernstein and H.K.T.Wong (1980), A Language Facility for Designing Database-Intensive Facilities, ACM Transactions on Database Systems **5** 185–207.

W.Reisig (1985), Petri Nets, an Introduction, Springer Verlag, Berlin.

B.N. Rossiter (1985), Full Text Data Base Management Systems: A Model and Implementation for Law, II International Congress Logica, Informatica, Diritto, edd. A.A.Martino, F. Socci Natali, 585–598.

B.N. Rossiter (1987), Machine Awareness in Database Technology, Proceedings Symposium VI, Meta-intelligence and the Cybernetics of Consciousness, XI International Congress of Cybernetics, Namur 1–9.

B.N.Rossiter and M.A.Heather (1990), Strengths and Weaknesses of Database Models for Textual Documents, Proceedings EP90, ed. R. Furuta, Cambridge 125–138.

B.N. Rossiter & M.A.Heather (1993), Database Architecture and Functional Dependencies Expressed with Formal Categories and Functors, University of Newcastle upon Tyne, Computing Science Technical Report no 432.

F.W.Tompa (1989), A Data Model for Flexible Hypertext Database Systems, ACM Transactions on Information Systems **7**(1) 85–106.

S.A.Treu, (1971), A Conceptual Framework for the Searcher-System Interface, in: Interactive Bibliographic Search: The User/Computer Interface, ed. D. E. Walker, AFIPS Press, Palo Alto 53–66.

D.C.Tsichritzis and O.M.Nierstrasz (1988), Fitting Round Objects into Square Data bases, ECOOP '88 Proceedings, in: Lecture Notes in Computer Science, Springer-Verlag **322** 283–299.

E.Yoder and T.C.Wettach (1989), Using Hypertext in a Law Firm, in: Hypertext'89 Proceedings, Special Issue - SIGCHI Bulletin 159–167.

P.T.Zellweger (1989), Scripted Documents: A Hypermedia Path Mechanism, in: Hypertext'89 Proceedings, Special Issue - SIGCHI Bulletin 1–14.