In ZUM'95: The Z Formal Specification Notation, J. Bowen and M. Hinchey, (eds.), Lecture Notes in Computer Science, 967, 83-102, Springer-Verlag, Heidelberg, 1995

Structuring Specification in Z to Build a Unifying Framework for Hypertext Systems

Mark d'Inverno and Mark Priestley

School of Computer Science, University of Westminster, 115 New Cavendish Street, London, W1M 8JS, UK. Email {dinverm,priestm}@westminster.ac.uk

Abstract. A report is given on work undertaken to produce a structured specification in Z of a model which aims to capture the essential abstractions of hypertext systems. The specification is presented in part and the potential value of this specification to the hypertext community is explored and discussed. We argue that this specification provides a framework for hypertext systems in that it provides: explicit and unambiguous definitions of hypertext terms, an explicit environment for the presentation, comparison and evaluation of hypertext systems and a foundation for future research and development in the field. Although there are many formal reference models of hypertext, we have found Z expressive enough to allow a unified account of a system and its operations. Our model does not restrict the specifier to any particular design, but provides a mathematical framework within which different models may be compared. Further, we were able to structure the specification in order that the model could be described initially at the highest level of abstraction with complexity added at increasingly lower levels of abstraction. This structured specification allows the functionality of hypertext systems to be considered at different levels of granularity which, we argue, gives rise to a well-defined robust model and a beneficial environment within which to reason about hypertext design. The use of this model in presenting and comparing existing models, as well as its use in developing a new hypertext learning model, is briefly discussed.

1 Introduction

Many formal reference models of hypertext have been presented in the literature [1, 12, 17, 20, 21], and whilst these models give valuable theoretical insights into certain aspects of the structure of hypertext, they are not by themselves adequate vehicles for the presentation, evaluation and comparison of different systems. In this paper we describe an approach to the formal specification of hypertext systems which allows the development of a common conceptual framework and provides an environment in which to discuss, design, develop and evaluate hypertext systems.

The language Z is based upon basic mathematical ideas. This means that it is accessible to many hypertext practitioners, and unlike many models [20, 21], is expressive enough to allow a consistent formal unified account of a system and its associated operations. We claim further that a well-structured Z specification built up from basic mathematical ideas can provide the following for the hypertext community:

Clarity. The use of formal concepts allows explicit and unambiguous descriptions of terms and complex systems to be given.

Common Conceptual Framework. The availability of explicit notations allows a movement from informal mutually inconsistent descriptions of systems towards a common understanding of the basic features and concerns of a particular class of systems.

Design Evaluation. This common framework enable alternative designs of particular systems to be explicitly presented, compared and evaluated.

Reliability. A formal specification language provides a proof system and a set of proof obligations which enables a reliable and robust model of systems to be built.

A well-structured specification in this context is one which first describes a system at its highest level of abstraction, with complexity added at each successive lower level of abstraction, allowing irrelevant information to be removed from consideration. Further, different modular components of a system can be isolated and described separately and commonalities in different parts of a system can be recognised and presented as such. Abstraction renders prejudice about design unnecessary; consequently, a specification of a "general system" can be written. Indeed, Z schema boxes have been ideal for manipulation in the design process since, viewing in many cases the design process as a constraint of possible states, design strategies can be presented as predicates in the appropriate state schemas. Such a structured specification, we argue, is a tool which enables a more considered hypertext analysis.

1.1 Motivation

This paper is a consequence of two separate ongoing areas of work. The first area involved the writing of programs to translate between various hypertext systems currently used at University of Westminster. In order to ensure that the structure of a document is preserved in its translation, it was decided to produce full formal specifications of the University of Westminster systems [9, 10]. This activity produced great benefits in understanding the differences and similarities between these systems and indicated that a suitable formal specification of a general hypertext system would be of great value in comparing different hypertext systems. The second major influence is the writing and presentation of formal specifications (in Z) with similar 'unifying' motivations in certain areas of computer science including the fields of interactive conferencing systems [8], distributed artificial intelligence [7] and multi-agent systems [18]. These papers demonstrate the need to provide further suitable formal specifications of complex systems in diverse application areas.

1.2 Related Work

There is another attempt in the hypertext literature to provide a formal specification of a 'general' hypertext system known as the Dexter Model of Hypertext [15]. Interestingly, the authors also chose Z with the motivation - to capture formally and informally the important abstractions found in a wide range of existing and future hypertext systems - similar in some respects to our own. The model essentially comprises a collection of components - links and nodes - with an accessor function which maps a unique identifier to a node and an resolver function which maps descriptions of components to the components themselves. The operations specified are that of adding, modifying and retrieving components.

However the specification is often obtuse and over-complicated, and only the most experienced Z practitioner with a good knowledge of hypertext would be able to gain much from the specification. Part of the problem is that no structuring of the specification takes place, rather it starts with a large collection of given sets and introduces many concepts and functions before the first state schema is actually presented. In this sense, the specification is very 'flat' and does not aid the reader in building up a picture of their model of a hypertext. Further, the specification describes hypertext at a very low level of detail and hence is much more orientated towards implementation concerns.

We argue that the immediate complexity will not serve the standard hypertext practitioner in providing an accessible model which can be commonly adopted by the hypertext community, and that the lack of abstraction mechanisms within the 'flat' specification does not provide a framework in which to present and develop ideas in the design of hypertext systems.

In addition, there is a system known as HAM - A General-Purpose Hypertext Abstract Machine [3], which is a general purpose server for a hypertext storage system. HAM has several general features similar to our own including nodes, links, graphs and attributes and describes the way information is represented before it is used in Information Retrieval. The motivation is essentially to lay the foundations for a standard terminology for the development of hypertext technology. Important as this work is, the model is only concerned with problems of storage, and not with representing an Information Retrieval session. In addition, the model is not formal, and subsequently does not provide the precision of a mathematical specification. Our work on the other hand is both formal and, we argue, sufficiently expressive to provide a framework within which to detail all aspects of hypertext.

1.3 An Overview of the Paper

The specification of our framework is split into two parts, and these are defined in sections 2 and 3. The first part, given in section 2, presents what we believe is the most straightforward and intuitive description of a model of hypertext systems where nodes are treated as given sets and links as a pair of system nodes. The second part, given in section 3, builds on the basic model and increases the level of specification detail in order to describe the internal details of a node. Each of these two sections is divided into three subsections: the first defines the structure of the system, the second the state of the system as it is being read and the third presents a description of the basic applications of hypertext, namely how it facilitates structured movement through a particular information space. Section 4 outlines how the model can be used to detail other features and applications of hypertext. Section 5 provides a summary of the paper and details current and future work.

2 The Basic Hypertext

2.1 Structure

If we consider a hypertext system at its highest level of abstraction, it consists of a collection of basic elements. These elements are typically called nodes, but the name can vary from system to system: for example, they are called *cards* in NoteCards [14], *frames* in KMS [2], *documents* in Intermedia [22], and *statements* in Augment [11]. In this specification we chose to use the word *node*, being the most common.

$[\mathsf{NODE}]$

CONTENTS

$Nodes : \mathbb{P}\,\mathsf{NODE}$

However, if we take a closer look at a hypertext system, we find that the structure is more sophisticated and that between nodes there exist certain connections, known as links, each suggesting some relationship between the nodes they connect. A link is directional, pointing from one node (sometimes referred to as the parent node) to another node (sometimes referred to as the child node). A link is therefore characterised by the nodes it connects.

$\mathsf{LINK} == \mathsf{NODE} \times \mathsf{NODE}$

LINKS

$Links : \mathbb{P}\,\mathsf{LINK}$

We define a hypertext system as a collection of nodes and links, where links must point *from* an existing system node. However, it is not the case that a link must necessarily point to a system node: many hypertext systems include the notion that some links only have the potential to point to such a node (e.g. [9]).

HYPERTEXT

CONTENTS
LINKS

$\mathsf{dom}\, Links \subseteq Nodes$

Button Nodes A more detailed investigation of a hypertext system will reveal that some nodes are special in that they may be reached without using a link. We call such nodes *button* nodes. Further, there may or may not be a default starting node when a hypertext system is first used in an IR session. (For definitions of optional and related concepts, please consult the Appendix).

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathsf{ButtonHYPERTEXT}} \\
\hline
\mathsf{HYPERTEXT} \\
Buttons : \mathbb{P}\,\mathsf{NODE} \\
StartNode : \mathsf{optional}\ [\mathsf{NODE}] \\
\hline
StartNode \subseteq Nodes \\
Buttons \subseteq Nodes \\
StartNode \subseteq Buttons \\
\hline
\end{array}$$

Typed Links In certain hypertext systems [9], links may be grouped into *Link Functions*.

$$\mathsf{LINKFUNCTION} == \mathsf{NODE} \nrightarrow \mathsf{NODE}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathsf{TypedLINKS}} \\
\hline
\mathsf{LINKS} \\
LinkFunctions : \mathbb{P}\,\mathsf{LINKFUNCTION} \\
\hline
\bigcup LinkFunctions \subseteq Links \\
\hline
\end{array}$$

A particular system might insist that a link could not belong to more than one function and that every link should belong to a function. In which case, we would simply include the following predicate in the above schema.

$$\mathsf{setdisjoint}\ LinkFunctions \wedge \bigcup LinkFunctions = Links$$

2.2 The State of the Hypertext

The next aspect of this simple hypertext model is to specify the state of the model as it is used in an Information Retrieval (IR) session. In all systems that we have investigated, there is a notion of the position of a user within the information space, and the history of that user's IR session. A history provides a record of the nodes visited by a user in a session, and possibly the way in which they were visited.

Our general model of a hypertext session history is, then, a set of sequences of nodes, where each of these sequences is updated in one of four ways depending on what type of move is made: the sequence can remain unchanged, the last-visited node can be appended, the sequence can be truncated at the first occurrence of the last-visited node, or fourthly, at the last occurrence of the last-visited node. For the sake of brevity, we do not present this mechanism in this paper, but for the purposes of exposition, we will show how two commonly-found browsing histories are used in IR. We define *StandardHistory* as a sequence of all the nodes visited, and *Visited* to be the set of nodes which have been visited.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textsf{HISTORY}} \\
\hline
\mathit{StandardHistory} : \mathsf{seq}\ \mathsf{NODE} \\
\mathit{Visited} : \mathbb{P}\ \mathsf{NODE} \\
\hline
\mathit{Visited} = \mathsf{ran}\ \mathit{StandardHistory} \\
\hline
\end{array}
$$

We represent a user session by the hypertext, their history and their current position within the information space.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textsf{HYPERTEXTState}} \\
\hline
\mathsf{HYPERTEXT} \\
\mathsf{HISTORY} \\
\mathit{CurrentNode} : \mathsf{NODE} \\
\hline
\mathit{CurrentNode} \in \mathit{Nodes} \\
\hline
\end{array}
$$

As mentioned, the buttons which become available during a session might be dependent on the session itself. Here the variable *RunButtons*, a superset of *Buttons*, represents those nodes which can currently be visited without the use of a link. In particular, it is typical that any previously visited node can be re-visited without using a link.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textsf{ButtonHYPERTEXTState}} \\
\hline
\mathsf{HYPERTEXTState} \\
\mathsf{ButtonHYPERTEXT} \\
\mathit{RunButtons} : \mathbb{P}\ \mathsf{NODE} \\
\hline
(\mathsf{defined}\ \mathit{StartNode}) \land (\mathit{Visited} \neq \{\}) \Rightarrow \\
\qquad\qquad\qquad \mathit{head}\ \mathit{StandardHistory} = \mathsf{the}\ \mathit{StartNode} \\
\mathit{Buttons} \subseteq \mathit{RunButtons} \\
\mathit{RunButtons} \subseteq \mathit{Nodes} \\
\mathit{Visited} \subseteq \mathit{RunButtons} \\
\hline
\end{array}
$$

2.3 Applications

One of the benefits of Z is that the operations that a system provides can be specified within the same formal framework. This property is not shared by many of the mathematical models presented in the literature; for example, [21] uses hypergraphs to give a formal account of the structure of a hypertext, but then specifies the operations of reading the hypertext using a mixture of pseudocode and informal English description. A particular advantage of a unified specification, as provided by Z, is that the properties of operations, and their effects on the state of the system, can be explored and reasoned about formally.

Any operation in an IR session will not alter the actual linked structure of the hypertext.

$$\begin{array}{|l}
\multicolumn{1}{l}{\Delta\mathsf{HYPERTEXTState}} \\ \hline
\mathsf{HYPERTEXTState} \\
\mathsf{HYPERTEXTState}' \\
\Xi\mathsf{HYPERTEXT} \\ \hline
\end{array}$$

Starting a hypertext session changes the state of the hypertext by resetting the history.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathsf{Login}} \\ \hline
\Delta\mathsf{HYPERTEXTState} \\ \hline
StandardHistory' = \langle\rangle \\ \hline
\end{array}$$

We now show how the hypertext is used in an IR session by moving through an information space using the hypertext system. Essentially a hypertext system supports two types of moves: first, a user may move from one node to another node by use of a link from their current node to another related node; second, if they have some knowledge of a node - because it is a button node or a previously visited node, for example - they may move directly to that node without using a link.

We structure the specification by first describing the properties that we want a general move operation to have before giving details of a particular move. This provides an example of how Z has enabled us to modularise this specification and so present the model in levels of increasing detail. In particular it shows how the small schemas defined in our specification can be combined to define more complex states and operations. In general, a move operation will change the state of the hypertext: the current node may alter and the history may alter, but the actual linked structure of the hypertext will remain unchanged. Further, a move operation may return a message to the user.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathsf{Move}} \\ \hline
\Delta\mathsf{HYPERTEXTState} \\
message! : \mathsf{optional}\ [\mathsf{ERROR}] \\ \hline
\end{array}$$

We can next distinguish between successful and unsuccessful attempts to move. A successful move will update the history list, appending the node that has just been visited. Further, there will be no message.

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathsf{MoveOk}} \\ \hline
\mathsf{Move} \\ \hline
Visited' = Visited \cup \{CurrentNode\} \\
StandardHistory' = StandardHistory \frown \langle CurrentNode\rangle \\
\mathsf{undefined}\ message! \\ \hline
\end{array}$$

Failed moves leave the hypertext state unchanged. An error message is given.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textsf{MoveFail}} \\
\textsf{Move} \\
\Xi\textsf{HYPERTEXTState} \\
\hline
\textsf{defined}\ message!
\end{array}$$

Now we describe using a link to move from the current node to another.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textsf{FollowLinkOk}} \\
link : \textsf{LINK} \\
\textsf{MoveOk} \\
\hline
CurrentNode' = second\ link
\end{array}$$

This move can be made in one of two ways. The user can select either the link to be used, or, a link function intended to isolate an appropriate link.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textsf{UserChosesLinkOk}} \\
\textsf{FollowLinkOk}[link?/link] \\
\hline
link? \in Links\ \wedge\ first\ link? = CurrentNode\ \wedge\ second\ link? \in Nodes
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\textsf{FollowLinkFunctionOk}} \\
LinkFunction? : \textsf{LINKFUNCTION} \\
\textsf{FollowLinkOk} \\
\textsf{TypedLINKS} \\
\hline
LinkFunction? \in LinkFunctions \\
CurrentNode \in (\textsf{dom}\ LinkFunction?) \\
link = (CurrentNode, LinkFunction?\ CurrentNode)
\end{array}$$

$\textsf{UserChoosesLinkFunctionOk} \mathrel{\widehat=} \textsf{FollowLinkFunctionOk} \setminus (link)$.

Moves can be made using buttons by supplying the node to be visited.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textsf{MoveButtonOk}} \\
\textsf{MoveOk} \\
\Delta\textsf{ButtonHYPERTEXTState} \\
\Xi\textsf{ButtonHYPERTEXT} \\
button? : \textsf{NODE} \\
\hline
button? \in RunButtons \\
CurrentNode' = button?
\end{array}$$

Moving back to the previously visited node requires no input node.

MoveBack

MoveOk

$StandardHistory \neq \langle\rangle$

$CurrentNode' = last\ StandardHistory$

The history may be up updated in a different way as follows. This justifies the need to model a more general history mechanism as mentioned previously.

MoveBackAlternative

Move

$StandardHistory \neq \langle\rangle$

undefined $message!$

$CurrentNode' = last\ StandardHistory$

$StandardHistory = front\ StandardHistory$

3 The Highlight Hypertext

We now lower the level of description of a hypertext system, in order to model the fact that a node contains certain hypertext elements. These elements are references, typically taking the form of highlighted text, which can then serve as the destination of, or source for, hypertext links. Many hypertext systems facilitate not only links connecting nodes, but regions within nodes.

3.1 Structure

Each node has a - possibly empty - set of internal hypertext references which we call highlights. Many hypertext systems support the operations of scrolling backwards and forwards through these highlights. Assuming that the highlights are unique within a node, we represent them using an injective sequence.

[HIGHLIGHT]

HighlightNODE

$Highlights : \mathrm{iseq}\ \mathsf{HIGHLIGHT}$

The inside of each node then consists of highlights.

HighlightNODES

CONTENTS

$GetHighlights : \mathsf{NODE} \nrightarrow \mathsf{HighlightNODE}$

$\mathrm{dom}\ GetHighlights = Nodes$

Next, we extend the notion of a link so that they can point to highlights within a node. We use the categories of Conklin [6] who differentiates two categories of link by drawing a distinction between *organisational* and *referential* links. We use these categories and introduce a third type which can be found in current hypertext systems [10] known as *span* links.

Organisational links Many hypertexts have an underlying structure, either as a consequence of the information space itself, or the way that the information space is required to be presented to a user. Organisational links capture this underlying structure. For example, these may be hierarchical in nature so that there might be a standard way within the hypertext of moving from a given node to a "parent", "child" or "sibling" node.

Referential links Referential links are typically non-hierarchical. They connect a *highlight*, which can be a point or a region within a node, to a another node. Referential links are motivated by the *content* of a node, rather than by the underlying structure of the hypertext or information space.

Span links We define Span links to be links which connect a *highlight* within a node to a *highlight* within another node. The notion of a cross-reference, for example, could be modelled in this fashion.

In this specification, we differentiate between these three types of link: we call organisational links *orglinks*, referential links *reflinks* and span links *spanlinks*. Some other mathematical models have had problems defining different kinds of link. In [13], this ability is described as an "innovative feature". In order to specify this more detailed hypertext, we must define a new type to represent links between highlights, and define each of the three link categories as subtypes.

HighlightLINK

$From, To : \mathsf{NODE}$

$FromHighlight, ToHighlight : \mathsf{optional}\ [\mathsf{HIGHLIGHT}]$

OrgLINK

HighlightLINK

$\mathsf{undefined}\ FromHighlight \land \mathsf{undefined}\ ToHighlight$

RefLINK

HighlightLINK

$\mathsf{defined}\ FromHighlight \land \mathsf{undefined}\ ToHighlight$

SpanLINK

HighlightLINK

$\mathsf{defined}\ FromHighlight \land \mathsf{defined}\ ToHighlight$

$(From \neq To) \land (FromHighlight \neq ToHighlight)$

We may wish to reason about these links in terms of the nodes they connect without concern about the kind of link they are. In order to do this we introduce a function which maps our new representation of links to our old representation.

$$
\begin{array}{|l}
\mathit{RecoverLink} : \mathsf{HighlightLINK} \twoheadrightarrow \mathsf{LINK} \\
\mathit{RecoverLinks} : \mathbb{P}\,\mathsf{HighlightLINK} \twoheadrightarrow \mathbb{P}\,\mathsf{LINK} \\
\hline
\forall\, c : \mathsf{HighlightLINK};\ cs : \mathbb{P}\,\mathsf{HighlightLINK} \bullet \\
\qquad\qquad \mathit{RecoverLink}\ c = (c.\mathit{From}, c.\mathit{To}) \wedge \\
\qquad\qquad \mathit{RecoverLinks}\ cs = \mathit{RecoverLink}(\!| cs |\!)
\end{array}
$$

We can now give the set of all links of the hypertext and relate the two representations of organisational link within the model.

$$
\begin{array}{|l}
\hline
\mathsf{HighlightLINKS} \\
\hline
\mathsf{LINKS} \\
\mathit{OrgLinks} : \mathbb{P}\,\mathsf{OrgLINK} \\
\mathit{RefLinks} : \mathbb{P}\,\mathsf{RefLINK} \\
\mathit{SpanLinks} : \mathbb{P}\,\mathsf{SpanLINK} \\
\mathit{HighlightLinks} : \mathbb{P}\,\mathsf{HighlightLINK} \\
\hline
\mathit{HighlightLinks} = \mathit{OrgLinks} \cup \mathit{RefLinks} \cup \mathit{SpanLinks} \\
\mathit{Links} = \mathit{RecoverLinks}\ \mathit{OrgLinks} \\
\hline
\end{array}
$$

Our new model of hypertext is then given by the following schema which ensures that all links are well defined.

$$
\begin{array}{|l}
\hline
\mathsf{HighlightHYPERTEXT} \\
\hline
\mathsf{HYPERTEXT} \\
\mathsf{HighlightLINKS} \\
\mathsf{HighlightNODES} \\
\hline
\forall\, l : \mathit{HighlightLinks} \bullet (l.\mathit{From} \in \mathit{Nodes}) \wedge \\
\qquad (l.\mathit{FromHighlight} \subseteq \operatorname{ran} (\mathit{GetHighlights}\ l.\mathit{From}).\mathit{Highlights}) \\
\hline
\end{array}
$$

Typing the links is similar to that given in the basic model, but the definition of what constitutes a link function is slightly different. A link function is any set of links for which no two links have the same ‘from-node’ and ‘from-highlight’. In other words, there is only one way to leave a given position given a particular link function. Further, we assert that a typed link function will only contain links which are either all organisational, referential or span.

$$
\begin{array}{l}
\mathsf{HlightLINKFUNCTION} == \{ xs : \mathbb{P}\,\mathsf{HighlightLINK};\ x, y : \mathsf{HighlightLINK} \mid \\
\qquad\qquad ((x \in xs) \wedge (y \in xs) \wedge (x \neq y)) \Rightarrow \\
\qquad (x.\mathit{From}, x.\mathit{FromHighlight}) \neq (y.\mathit{From}, y.\mathit{FromHighlight}) \bullet xs \}
\end{array}
$$

TypedHighlightLINKS

HighlightLINKS
$OrgLinkFuns, RefLinkFuns, SpanLinkFuns : \mathbb{P}\ \mathsf{HlightLINKFUNCTION}$

$\bigcup OrgLinkFuns \subseteq OrgLinks$
$\bigcup RefLinkFuns \subseteq RefLinks$
$\bigcup SpanLinkFuns \subseteq SpanLinks$

3.2 State

We now define the position of a user within the hypertext. This will not only include the current node and history from the state of the basic model, but also the position of a user within a node. This position will either be defined, in which case the user will be positioned at some highlight, or undefined, which occurs, for example, when a node has no highlights or an organisational link has just been used to move to the current node.

HighlightHYPERTEXTState

HYPERTEXTState
HighlightHYPERTEXT
$Position : \mathsf{optional}\ [\mathsf{HIGHLIGHT}]$
HighlightNODE

$\theta\mathsf{HighlightNODE} = GetHighlights\ CurrentNode$
$Position \subseteq (\mathsf{ran}\ Highlights)$

3.3 Applications

A change in the state will not affect the structure.

ΔHighlightHYPERTEXTState

HighlightHYPERTEXTState
HighlightHYPERTEXTState$'$
ΞHighlightHYPERTEXT

Any move using highlights may affect the state.

HighlightMove

ΔHighlightHYPERTEXTState
Move

We distinguish between two types of move. Internal moves involve the user scrolling through or selecting one of the highlights of the current node. External moves, on the other hand, involve the user taking a link, or moving to a button node. An internal move will not affect the state of the basic hypertext - the current node and history are not altered - and can only be made if the current node actually contains highlights.

InternalMoveOk

HighlightMove
ΞHYPERTEXTState
MoveOk

$Highlights \neq \langle\rangle$

There are three basic internal moves within a node: moving to the next highlight, moving to the previous highlight - both which necessarily entail that the current position is defined - and moving to a chosen highlight. For a definition of CycleNext and CyclePrevious please see the appendix.

NextHighlight

InternalMoveOk

defined $Position$
the $Position' = \mathsf{CycleNext}\ ((\mathsf{the}\ Position), Highlights)$

PreviousHighlight

InternalMoveOk

defined $Position$
the $Position' = \mathsf{CyclePrevious}\ ((\mathsf{the}\ Position), Highlights)$

SelectHighlight

InternalMoveOk
$highlight? : \mathsf{HIGHLIGHT}$

$highlight? \in (\mathsf{ran}\ Highlights)$
the $Position' = highlight?$

The general external move may affect the position and the current node. Note that the use of buttons is not changed in any way in this more sophisticated model.

ExternalMoveOk

HighlightMove
MoveOk
ΞHighlightHYPERTEXT

To use a link successfully, the parent node of the link must necessarily be the current node. If we use an organisational link then this is sufficient; they can be used from any position within a node. However, if we use a span or referential link then the link must have a from-highlight equal to the current position. We re-use our basic model definition as follows:

$$\begin{array}{|l}
\multicolumn{1}{l}{\textsf{FollowHighlightLinkOk}} \\
\hline
\textsf{ExternalMoveOk} \\
\textsf{FollowLinkOk} \\
\mathit{highlightlink} : \textsf{OrgLINK} \\
\hline
\mathit{highlightlink}.\mathit{From} = \mathit{CurrentNode} \\
\mathit{highlightlink} \in \mathit{OrgLinks} \lor (\mathit{highlightlink} \in (\mathit{RefLinks} \cup \mathit{SpanLinks}) \\
\qquad\qquad\qquad\qquad \land\ \mathit{Position} = \mathit{highlightlink}.\mathit{FromHighlight}) \\
\mathit{link} = (\mathit{highlightlink}.\mathit{From}, \mathit{highlightlink}.\mathit{To}) \\
\mathit{Position}' = \mathit{highlightlink}.\mathit{ToHighlight} \\
\hline
\end{array}$$

$$\textsf{UserChoosesHighlightLinkOk} \ \widehat{=}\ \textsf{FollowHighlightLinkOk} \setminus (\mathit{link})$$

Again the user either chooses the link, or a link function intended to isolate the required link.

4 Extensions

Although many formal models of hypertext have been proposed in the literature, there is still little consensus about what a definitive model should be. Indeed one might argue that, in view of the rapid progress of the technology, it is probably premature to attempt a definitive formalisation. However, a significant benefit of Z that we have discovered in this respect is that it does not restrict the specifier to any particular mathematical model; rather it provides a general mathematical framework within which different models, and even particular systems, can be defined and contrasted.

We now justify this claim here by considering a number of more sophisticated features of hypertext and show how the model defined in the previous sections can be elaborated to define and describe features and extensions found in a variety of hypertext systems.

4.1 Values and Attributes

A node or link may have a collection of types with associated values which may be used to structure the state space. In this case, the hypertext can be structured so that only certain types of links have access to certain types of nodes. In the following schema, we assert that for any two nodes connected by a link, the link and the nodes must have an associated type in common.

$$[\textsf{TYPE}, \textsf{VALUE}]$$
$$\textsf{TYPEVALUEPAIRS} == \mathbb{P}(\textsf{TYPE} \times \textsf{VALUE})$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\textsf{TypedHYPERTEXT}} \\
\textsf{HighlightHYPERTEXT} \\
NodeType : \textsf{NODE} \longrightarrow \textsf{TYPEVALUEPAIRS} \\
HighlightLinkTypes : \textsf{HighlightLINK} \longrightarrow \textsf{TYPEVALUEPAIRS} \\
\hline
\textsf{dom}\, NodeType = Nodes \\
\textsf{dom}\, HighlightLinkTypes = HighlightLinks \\
\forall c : HighlightLinks \mid c.To \in Nodes \bullet \\
\quad (\exists t : \textsf{TYPE} \bullet (t \in \bigcap\{first \llparenthesis HighlightLinkTypes\ c \rrparenthesis, \\
\qquad\qquad first \llparenthesis NodeType\,(c.To) \rrparenthesis, first \llparenthesis NodeType\,(c.From) \rrparenthesis\})) \\
\hline
\end{array}$$

4.2 Specifying Different Topologies

The *topology* of a hypertext describes the way in which the nodes are connected. In the simplest case, a hypertext is seen simply as being a directed graph, but other organisations are possible to facilitate the successful movement though an information space. In [19] a survey is given of possible topologies and, as an example, we define a hypertext which has a *hierarchical* structure. This is defined in terms of the organisational link function *Parent* and a set of organisational link functions called *Children*.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textsf{HierarchicalHYPERTEXT}} \\
\textsf{HighlightHYPERTEXT} \\
\textsf{TypedHighlightLINKS} \\
\textsf{ButtonHYPERTEXT} \\
Parent : \textsf{HlightLINKFUNCTION} \\
Children : \mathbb{P}\ \textsf{HlightLINKFUNCTION} \\
\hline
Parent \in OrgLinkFuns \\
Children \subseteq OrgLinkFuns \\
\textsf{defined}\ StartNode \\
\textsf{ran}\,(RecoverLinks\,(\bigcup Children)) = Nodes \setminus StartNode \\
(RecoverLinks\ Parent)^{\sim} = RecoverLinks\,(\bigcup Children) \\
\hline
\end{array}$$

4.3 User Navigation

As a mechanism to help the user navigate through a document, a number of proposals have been made for defining *paths* through the document [20]. A path would offer a reader a pre-defined route through a subset of the document, thus enabling an overview of the hypertext, or of a particular subject to be presented. A path may be just a set of nodes. Equally it may take the form of a sequence of links which actually take the user through particular highlights in the hypertext.

$$\textsf{SIMPLEPATH} == \textsf{seq NODE}$$
$$\textsf{PATH} == \textsf{seq HighlightLINK}$$

$$\begin{array}{|l}
\text{\_\_SimplePATHS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
\mathsf{CONTENTS} \\
\mathit{SimplePaths} : \mathbb{P}\, \mathsf{SIMPLEPATH} \\
\hline
\mathsf{ran}\,(\bigcup \mathit{SimplePaths}) \subseteq \mathit{Nodes} \\
\hline
\end{array}$$

$$\begin{array}{|l}
\text{\_\_PATHS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
\mathsf{HighlightLINKS} \\
\mathit{paths} : \mathbb{P}\, \mathsf{PATH} \\
\hline
\forall\, p : \mathit{paths} \bullet (\forall\, l, m : \mathsf{HighlightLINK} \mid \langle l, m \rangle \;\mathsf{in}\; p \bullet \\
\qquad\qquad\qquad (l.\mathit{To}, l.\mathit{ToHighlight}) = (m.\mathit{From}, m.\mathit{ToHighlight})) \\
\hline
\end{array}$$

4.4 Content of Nodes

In this specification we have not considered the text that is stored at each node. In the following, *Text* is defined as a schema since we wish to allow for the possibility of different nodes sharing the same content, cited as an advantage of the hypergraph model in [21].

$$\begin{array}{l}
[\mathsf{CHAR}] \\
\mathsf{STRING} == \mathsf{seq}\, \mathsf{CHAR} \\
\mathsf{TEXT} \mathrel{\widehat{=}} [\mathit{text} : \mathsf{STRING}]
\end{array}$$

$$\begin{array}{|l}
\text{\_\_TextHYPERXTET\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
\Xi \mathsf{HYPERTEXT} \\
\mathit{Text} : \mathsf{NODE} \mathrel{+\!\!\!\rightarrow} \mathsf{TEXT} \\
\hline
\mathsf{dom}\, \mathit{Text} = \mathit{Nodes} \\
\hline
\end{array}$$

Once a notion of content has been defined, it is possible to define a further class of move operations which Conklin [6] calls *keyword links*. A simple example would be to search for all nodes containing a given string. The schema which describes the successful operation is given below. The input, *keyword*?, is the search string and the set of nodes in the document containing the string is returned in the set *found*!. One of these nodes may become the current node.

$$\begin{array}{|l}
\text{\_\_KeywordSearch\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
\mathsf{TextHYPERXTET} \\
\mathit{keyword}? : \mathsf{STRING} \\
\mathit{found}! : \mathbb{P}\, \mathsf{NODE} \\
\hline
\mathit{found}! = \{ n : \mathit{Nodes} \mid \mathit{keyword}? \;\mathsf{in}\; (\mathit{Text}\; n).\mathit{text} \bullet n \} \\
\hline
\end{array}$$

4.5 Properties of Hypertext

It is straightforward in the Z framework to define additional properties of hypertext. We give two examples here. The first states that every node in the hypertext can be reached from the start node of a user session (whether user or system defined) using organisational links only. The second that no nodes are dead ends, or equivalently, that there is always at least one organisational, referential or span link out of any node.

$$\begin{array}{|l}
\textsf{Accessibility} \\
\hline
\textsf{ButtonHYPERTEXT} \\
\hline
\textsf{defined } StartNode \Rightarrow Nodes \subseteq \textsf{ran}\,(StartNode \lhd Links^*) \\
\textsf{undefined } StartNode \Rightarrow (\forall\, n : Nodes \bullet Nodes \subseteq \textsf{ran}\,(\{n\} \lhd Links^*)) \\
\hline
\end{array}$$

$$\begin{array}{|l}
\textsf{NoDeadEnds} \\
\hline
\textsf{HighlightHYPERTEXT} \\
\textsf{TypedHighlightLINKS} \\
\hline
Nodes \subseteq \textsf{dom}\,(RecoverLinks\,(OrgLinks \cup RefLinks \cup SpanLinks)) \\
\hline
\end{array}$$

4.6 Authoring

In addition to operations for reading an information space using hypertext, many systems also provide *authoring* operations by means of which nodes and links can be added to a hypertext to represent the information space in a more effective way. Adding a new node is known as indexing, adding a new link is known as hyperization. In the following schema we describe the hyperization operation of creating a new organisational link in the highlight hypertext.

$$\begin{array}{|l}
\textsf{AddHighlightLink} \\
\hline
\Delta\textsf{HighlightHYPERTEXT} \\
hlink? : \textsf{HighlightLINK} \\
\hline
hlink? \notin HighlightLinks \\
hlink?.From \in Nodes \\
hlink? \in \textsf{OrgLINK} \\
OrgLinks' = OrgLinks \cup \{hlink?\} \\
RefLinks' = RefLinks \\
SpanLinks' = SpanLinks \\
\hline
\end{array}$$

As a further example, we define the operation of removing a node from the basic model where any links which point to or from that node are also deleted.

RemoveNode

$node? : \mathsf{NODE}$
$\Delta \mathsf{HYPERTEXT}$

$node? \in Nodes$
$Nodes' = Nodes \setminus \{node?\}$
$Links' = \{node?\} \ntriangleleft Links \ntriangleright \{node?\}$

5 Summary and Future Work

In this paper we have presented part of the formal specification we have developed in order to define a framework for hypertext systems. This, we argue, provides an environment in which to discuss, design, develop and evaluate such systems. Z has enabled us to produce a well structured specification accessible to researchers from a non-formal background. We have been able to describe hypertext at the highest level of abstraction and then, using increasingly detailed specification, we have been able to add necessary system complexity at appropriate levels. We have shown how constructing Z in such a way does not restrict a specifier to any particular mathematical model; rather it provides a general mathematical framework within which different models, and even particular systems, can be defined and contrasted.

The framework has provided the foundation upon which to build a formal model of a new learning system in hypertext [16]. This system uses statistical information collected in Information Retrieval sessions over a period of time to learn how best to aid the user in navigating through a given information space. Since our framework specification is well structured, we were able to choose the appropriate level of abstraction relevant to our purpose of modelling this new system. In this case, the learning model is concerned only with organisational links between nodes, and so we have developed the formalisms of the learning techniques within our most basic model of hypertext. Next we intend to formalise certain hypertext systems within this framework in order that the new learning model of hypertext may be incorporated into these systems.

Further work continues in developing a more general and generic version of the specification and in applying it to existing hypertext systems. We have specified several existing hypertext systems including HyperCard [4] and the World Wide Web [5] within our framework. Further, using the specifications of the two systems used by Westminster University [9, 10], we are now able to investigate a means of providing automatic translation rules between the two.

Acknowledgements

Thanks to Claire Cohen, Jennifer Goodwin, Paul Howells, Michael Hu and Michael Luck and the anonymous referees for comments on earlier versions of this paper. The specification contained in this paper was checked for correctness using the *f*UZZ package.

References

1. F. Afrati and C. Koutras. A hypertext model supporting query mechanisms. In *Hypertext: Concepts, Systems and Applications. Proceedings of the European Conference on Hypertext*, pages 52–66, 1990.
2. R. Akscyn, D. McCracken, and E. Yoder. KMS: A distributed hypertext for managing knowledge in organizations. *Communications of the ACM*, 31(7), July 1988.
3. B. Campell and J. Goodman. HAM: A General Purpose Hypertext Abstract Machine. *Communications of the ACM*, 31(7):856–861, 1988.
4. C. Cohen, M. d'Inverno, and M. Priestley. Z Specification of HyperCard. Technical report, Software Engineering Division, University of Westminster, 1995.
5. C. Cohen, M. d'Inverno, and M. Priestley. Z Specification of the World Wide Web. Technical report, Software Engineering Division, University of Westminster, 1995.
6. J. Conklin. Hypertext: An Introduction and Survey. *Computer*, 20(9):17–41, 1987.
7. M. d'Inverno. Using Z to capture the essence of a contract net. Master's thesis, Programming Research Group, Oxford University, 1988.
8. M. d'Inverno and J. Crowcroft. Design, specification and implementation of an interactive conferencing system. In *Proceedings of IEEE Infocom, Miami, USA. Published IEEE*, 1991.
9. M. d'Inverno and M. Priestley. Z specification of GNU Emacs info system. Technical report, Software Engineering Division, University of Westminster, 1994.
10. M. d'Inverno and M. Priestley. Z specification of IDEAs system. Technical report, Software Engineering Division, University of Westminster, 1994.
11. D. Englebart. Authorship provisions in Augment. In *Proceedings of the IEEE COMPCON*, Spring 1984.
12. P. Garg. Abstraction Mechanisms in Hypertext. *Communications of the ACM*, 31(7):862–870, 1988.
13. F. Garzotto, P. Paolini, and D. Schwabe. HDM–A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, 11(1):27–50, 1993.
14. F. Halasz. Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM*, 31(7), July 1988.
15. F. Halasz and M. Schwartz. The Dexter Hypertext. *Communications of the ACM*, 37(2):30–39, 1994.
16. M. J. Hu. *An Intelligent Information System.* PhD thesis, UCL, 1994.
17. D. Lange. A formal model for hypertext. In *Proceedings of the NIST Hypertext Standardization Workshop*, 1990.
18. M. Luck and M. d'Inverno. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 254–260. AAAI Press / MIT Press, 1995.
19. H. Van Dyke Parunak. Hypermedia Topologies and User Navigation. In *Hypertext '89 Proceedings*, 1989.
20. D. Stotts and R. Furata. Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. *ACM Transactions on Information Systems*, 7(1):3–29, 1989.
21. F. Tompa. A Data Model For Flexible Hypertext Database Systems. *ACM Transactions on Information Systems*, 7(1):85–100, 1989.
22. N. Yankelovich, B. Haan, N. Meyrowitz, and S. Drucker. Intermedia: The concept and the construction of a seamless information environment. *IEEE Computer*, 1988.

Appendix: Z Extensions

We have found it useful in a specification to be able to assert that an element is optional. For example, in the specification given in this paper, the error message returned by a hypertext move is optional. If the move is unsuccessful, there is an error message, if the move is successful then there is no error message. The following definitions provide for a new type, $\mathsf{optional}\ T$, for any existing type, T, along with the predicates $\mathsf{defined}$ and $\mathsf{undefined}$ which test whether an element of $\mathsf{optional}\ T$ is defined or not. The function, the, extracts the element from a defined member of $\mathsf{optional}\ T$. We further define a prefix relation, $\mathsf{setdisjoint}$, which holds for a set of sets if all the members of that set of sets are pairwise disjoint. Lastly we define two functions which cycle forwards and backwards through non-empty injective sequences.

$$\mathsf{optional}\,[X] == \{xs : \mathbb{P}\,X \mid \#\,xs \leq 1\}$$

$$\begin{array}{|l}
[X] \\
\hline
\mathsf{defined}\ \_, \mathsf{undefined}\ \_ : \mathbb{P}(\mathsf{optional}\,[X]) \\
\mathsf{the} : \mathsf{optional}\,[X] \rightarrowtail\!\!\!\!+ X \\
\hline
\forall\, xs : \mathsf{optional}\,[X] \bullet \mathsf{defined}\ xs \Leftrightarrow \#\,xs = 1 \land \\
\qquad\qquad\qquad\qquad \mathsf{undefined}\ xs \Leftrightarrow \#\,xs = 0 \\
\forall\, xs : \mathsf{optional}\,[X] \mid \mathsf{defined}\ xs \bullet \\
\qquad\qquad\qquad\qquad \mathsf{the}\ xs = (\mu\, x : X \mid x \in xs) \\
\hline
\end{array}$$

$$\begin{array}{|l}
[X] \\
\hline
\mathsf{setdisjoint}\ \_ : \mathbb{P}(\mathbb{P}(\mathbb{P}\,X)) \\
\hline
\forall\, xss : \mathbb{P}(\mathbb{P}\,X) \bullet \mathsf{setdisjoint}\ xss \Leftrightarrow (\forall\, xs, ys : \mathbb{P}\,X \bullet \\
\qquad\qquad ((xs \in xss) \land (ys \in xss) \land (xs \neq ys)) \Rightarrow (xs \cap ys) = \varnothing) \\
\hline
\end{array}$$

$$\begin{array}{|l}
[X] \\
\hline
\mathsf{CycleNext}, \mathsf{CyclePrevious} : (X \times \mathsf{iseq}\,X) \rightarrowtail\!\!\!\!+ X \\
\mathsf{Index} : (X \times \mathsf{iseq}\,X) \rightarrowtail\!\!\!\!+ \mathbb{N} \\
\hline
\forall\, s : \mathsf{iseq}\,X;\ x : X \mid x \in (\mathsf{ran}\ s) \bullet \mathsf{Index}\,(x,s) = s^{\sim}x \land \\
\quad \mathsf{Index}\,(x,s) \neq \#s \Rightarrow \mathsf{CycleNext}\,(x,s) = s(\mathsf{Index}\,(x,s) + 1) \land \\
\quad \mathsf{Index}\,(x,s) = \#s \Rightarrow \mathsf{CycleNext}\,(x,s) = \mathit{head}\ s \land \\
\quad \mathsf{Index}\,(x,s) \neq 1 \Rightarrow \mathsf{CyclePrevious}\,(x,s) = s\,(\mathsf{Index}\,(x,s) - 1) \land \\
\quad \mathsf{Index}\,(x,s) = 1 \Rightarrow \mathsf{CyclePrevious}\,(x,s) = \mathit{last}\ s \\
\hline
\end{array}$$