On the Expressiveness of Links in Hypertext Systems

LUC MOREAU AND WENDY HALL

Department of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK
Email: L.Moreau,wh@ecs.soton.ac.uk

In this paper, we study how linking mechanisms contribute to the expressiveness of hypertext systems. For this purpose, we formalise hypertext systems as abstract machines. As the primary benefit of hypertext systems is to be able to read documents non-linearly, their expressiveness is defined in terms of the ability to follow links. Then, we classify hypertext systems according to the power of the underlying automaton. The model allow us to compare embedded vs separate links and simple vs generic links. Then, we investigate history mechanisms, adaptive hypertexts, and functional links. Our conclusion is that simple links, whether embbeded or separate, generic links, and some adaptive links all give hypertext systems the power of finite state automata. The history mechanism confers them the power of pushdown automata, whereas the general functional links give them Turing completeness.

Keywords: links, generic, simple, embedded, separate, functional, expressiveness

Received month date, year; revised month date, year

1. INTRODUCTION

A hypertext system is a software system for creating, editing and browsing hyperdocuments [2]. Links represent logical and structural associations between documents; they allow a non-linear reading of documents [3], according to the user's needs or wishes. Many authors have argued that cross-referencing, i.e. linking, is an essential technique to facilitate access to information [4, 5].

The hypertext research community has defined many kinds of links and has proposed many ways of implementing them. In particular, a much debated issue has been whether links should be part of documents, e.g. in HTML [6], or whether they should be kept separate, e.g. in Intermedia [7]. A recent summary of the issue is P.J. Brown and H. Brown's paper "*Embedded or separate hypertext mark-up: is it an issue?*" [8]. Using practical considerations, they discuss the benefits or inconveniences of adopting one approach or the other; they also present an implementation able to combine both of them. However, they do not present a formal framework according to which systems can be judged on the basis of established criteria. Their article gave the initial impetus to the research described in this paper: a formal comparison of the power or expressiveness of hypertext systems.

The literature abounds in more or less formal models of hypertexts, but they have quite different motivations. Let us mention some of them. The Dexter model [9] attempts to define a common vocabulary and associated meaning in order to talk about hypertext systems. The hypertext abstract machine (HAM) [10] is an architectural description of a general-purpose, transaction-based, multi-user server for a hypertext storage system. The Trellis model [11] is a formal semantic specification of a hypertext based on petri nets. The same authors have regarded hypertexts as automata and have used model checking to prove some properties of a given hypertext [12]. Hypertexts have also been formalised as graphs [13]. However, to the best of our knowledge, no model is used to compare the expressive power of hypertext systems.

The physics and programming language communities use a "black box" approach to compare systems: a set of observable events is used to compare systems, independently of their internal behaviour. We adopt a similar approach in this paper: we model hyperdocuments and hypertext systems as abstract machines, i.e. kinds of automata, and we compare them according to a set of observable events. As others [12, 14], we study the dynamic properties of hypertexts in terms of "reader's experience". At this point, let us just state that we adopt the capability of following links as an observable event of a hypertext.

Having developed a formal way to compare hypertext systems, we first use it to answer Brown and Brown's question about embedded and separate links: as we might have expected, our conclusion does not differ from theirs. We however do not stop our investigation on this issue: we use our model to compare other forms of linking, and to study how they may change the expressiveness of hypertext systems. First, we study Microcosm's generic links [15, 16, 17] that are generally regarded as a powerful way to author hypertexts; we see how this power can be described formally. Then, we introduce navigation history in our semantics, and show how it can contribute to expressiveness. Next, we discuss adaptive hypertexts where the availability of a given link may vary over time or navigation history. Finally, we discuss a very general class of linking mechanism called the functional link [18].

This paper is organised as follows. In Section 2, we define a hypertext machine able to perform transitions in the same way as a user can navigate a hyperdocument. In our initial model, we use separate links. We also define a set of observable events according to which the expressiveness of hypertext systems can be compared. In Section 3, we present a formal account of Brown and Brown's discussion on embedded and separate links. Then, we successively investigate other mechanisms, such as generic links in Section 4, history mechanism in Section 5, and adaptive linking in Section 6. Finally, in Section 7, we integrate Ashman and Verbyla's [18, 19] notion of functional link into our model, and define its expressiveness: it is shown to be the most general form of linking, as it is able to emulate a Turing machine; as a result, this allows us to claim that *linking is as powerful as computing*.

2. A HYPERTEXT MACHINE

Following Bieber and Kimbrough [2], we define a *hypertext system* as the software for creating, editing, and browsing hyperdocuments. In this section, we introduce a *hypertext machine* as the formalisation of a hypertext system and hyperdocuments, which we shall also refer to as the semantics of hypertext. The semantics of hypertext is expressed in terms of the reader's experience, i.e. in terms of the navigation capability allowed by a hypertext system for a given hyperdocument. Let us note here and now that our semantics does not formalise the authoring and editing activities.

2.1. Definition

In physics, in order to compare properties of different systems, it is common practice to adopt a set of *observables* or *observable events* against which systems may be compared. Such an approach is also followed to compare the expressiveness of programming languages [20, 21]. Given a set of observables, two systems that cannot be distinguished by an observer are said to be *observationally equivalent*. For example, in order to compare two programming languages, the set of observables may be defined as the set of final results returned by programs.

We follow a similar approach for hypertext systems. The key characteristic of hypertexts is the existence of links between documents, and, as a result, their ability to provide for non-linear reading of documents [3]. Therefore, as far as end users are concerned, a valuable set of hypertext observables is the various *links* that they may traverse. We shall say that a user cannot distinguish two hypertext systems if all the sequences of links that may be traversed with one may be traversed with the other, and vice-versa, and if they give access to the same information. Indistinguishable hypertext systems will be said to have the *same expressiveness* because they have the ability to link documents in the same way.

Before presenting the formal framework, we define the following mathematical symbols:

$DOM(f)$	:	Domain of a function f
$A \rightarrow_f B$	:	Set of Finite Function with Domain A and Range B
$\mathbb{F}(A)$	:	Set of Finite Subsets

For the purpose of this paper, we choose a very abstract representation of documents. We regard a *document* as a finite function mapping *offsets* to *tokens* (cf. Figure 1). Tokens may be understood as characters or words; they are atomic objects containing information. Tokens appear at a given offset in the document. Both tokens and offsets are not further specified so that this definition remains valid for any type of media. For instance, in images, tokens could be pixels and the finite function would be a two-dimensional array. Several document representations use a tree organisation, for instance in structured text formats [22] or image formats [23]; offsets then become access paths to tree subcomponents.

We also assume that documents are named and that a *naming function* is able to map names to documents. Names belong to an unspecified set of names. In the context of the WWW, URLs are the equivalent of our names, and the naming function would be implemented by http servers, which return a document from the file system (or generated by a computation) in response to a URL. In their object-oriented description of hypermedia components, Gronbaek and Trigg [24] use ids to identify nodes; a similar mechanism is also used in the Dexter model [9].

A *link* is a pair composed of two *anchors*, respectively called *source* and *destination*. We are taking the definition that an anchor is the mechanism for referring to a part of a document [9]. In our abstract model, an anchor is defined as a pair composed of a document name and an offset; intuitively, an anchor refers to a token in a document. We have adopted a unidirectional notion of link, where the source anchor designates the location from which the link departs, whereas the destination anchor points at the location where the link arrives. Finally, a *linkbase* is defined as a set of links. For the time being, we only consider unidirectional links and we also assume that all links are point to point; we will investigate how other variants of links contribute

$$\begin{array}{lcll}
o \in \mathit{Offset} & = & \{m \in \mathbb{N}, m \geq 0\} & \text{(Offset)} \\
t \in \mathit{Token} & = & \{t_1, t_2, \ldots\} & \text{(Token)} \\
N \in \mathit{Name} & = & \{N_1, N_2, \ldots\} & \text{(Document Name)} \\
\\
D \in \mathit{Doc} & = & \mathit{Offset} \rightarrow_f \mathit{Token} & \text{(Document)} \\
\rho \in \mathit{Env} & = & \mathit{Name} \rightarrow_f \mathit{Doc} & \text{(Naming Function)} \\
\mathit{LB} \in \mathit{LBSet} & = & \mathbb{F}(\mathit{Link}) & \text{(LinkBase)} \\
D^* \in \mathit{DSet} & = & \mathbb{F}(\mathit{Doc}) & \text{(Set of Documents)} \\
\\
l \in \mathit{Link} & = & \mathit{Source} \times \mathit{Dest} & \text{(Link)} \\
s \in \mathit{Source} & = & \mathit{Name} \times \mathit{Offset} & \text{(Source Anchor)} \\
d \in \mathit{Dest} & = & \mathit{Name} \times \mathit{Offset} & \text{(Destination Anchor)} \\
S \in \mathit{State} & = & \mathit{Doc} \times \mathit{LBSet} \times \mathit{Env} \times \mathit{DSet} & \text{(State)}
\end{array}$$

Notation:

$$\begin{array}{rcll}
l & ::= & \langle s, d \rangle & \text{(Link)} \\
s & ::= & \langle N, o \rangle & \text{(Source Anchor)} \\
d & ::= & \langle N, o \rangle & \text{(Destination Anchor)} \\
S & ::= & \mathcal{H}_\mathsf{s}\langle D, LB, \rho, D^* \rangle & \text{(State)}
\end{array}$$

Link traversal relation:

$$\begin{array}{l}
\mapsto \;\subseteq\; \mathit{State} \times \mathit{State} \times \mathit{Link} \\
\mathcal{H}_\mathsf{s}\langle D_1, LB, \rho, D^* \rangle \mapsto^l \mathcal{H}_\mathsf{s}\langle D_2, LB, \rho, D^* \rangle \\
\quad \text{if} \quad l = \langle\langle N_1, o_1 \rangle, \langle N_2, o_2 \rangle\rangle \in LB, \text{ such that } \begin{array}[t]{l} \mathit{valid}(l, \rho, D^*) \\ \rho(N_1) = D_1, \\ \rho(N_2) = D_2. \end{array}
\end{array}$$

FIGURE 1. State Space and Transition Relation of a Hypertext Machine with Separate Links

to the expressiveness of hypertext systems later in the paper.

Notation We use the following notations. An instance of a link is represented by a pair $\langle s, d \rangle$, where s and d are source and destination anchors, themselves represented by a pair $\langle N, o \rangle$. An instance of a hypertext with separate links is noted by the quadruple $\mathcal{H}_\mathsf{s}\langle D, LB, \rho, D^* \rangle$, where the tag $\mathcal{H}_\mathsf{s}$ indicates a hypertext with *separate* links.

In order to specify the operational semantics of a hypertext machine, we need to define a notion of *link validity*. Intuitively, a link is valid if it is not dangling, i.e. if its source and destination anchors point at existing documents and offsets.

DEFINITION 1. A link $l = \langle\langle N_1, o_1 \rangle, \langle N_2, o_2 \rangle\rangle$ is *valid* with respect to a naming function ρ and a set of documents D^*, written $\mathit{valid}(l, \rho, D^*)$, if the following conditions hold, assuming that $D_1 = \rho(N_1)$ and $D_2 = \rho(N_2)$.

1. $D_1 \in D^*$,
2. $D_2 \in D^*$,
3. $o_1 \in \mathit{DOM}(D_1)$, and
4. $o_2 \in \mathit{DOM}(D_2)$.

A linkbase is valid with respect to a naming function ρ and a set D^* of documents if each of its links is valid with respect to D^* and ρ. □

For the purpose of conciseness, we say that links or linkbases are valid, without specifying the naming function or the set of documents they relate to.

Having specified the various entities of a hypertext machine, i.e. its state space, we can define the transitions that this *abstract machine* is able to perform. Transitions, which allow the machine to evolve from one *state* to another, mimic the *traversal* process. States are defined by triples composed of: *(i)* the document that is the current focus, i.e. the document that the user is currently reading, *(ii)* a current linkbase, *(iii)* a set of documents. The transition relation between states, called the *link traversal* relation, written $\mapsto$, also appears in Figure 1. There is a transition from state 1 to state 2, if there is a valid link in the current linkbase between the documents in focus in states 1 and 2. The reflexive, transitive closure of $\mapsto$ is written $\mapsto^*$.

Remark In this abstract machine, we do not specify how links are displayed to the user, nor do we take into account any superficial interface characteristics. Active links in a document may be highlighted by the system, or a user might select a word in order to follow a potential link. For the purpose of our expressiveness comparison, what really matters to us is the user's ability to follow a link, as modelled by the link traversal relation.

Given an initial state $S_i = \mathcal{H}_\mathsf{s}\langle D_i, LB, \rho, D^* \rangle$, *the set* *of reachable documents* from S_i:

$$reachable(S_i) = \{D \mid S_i \mapsto^* \mathcal{H}_s\langle D, LB, \rho, D^*\rangle\}$$

The essential feature of hypertexts is their ability to offer links between (parts of) documents, and henceforth to allow users to read documents non-linearly. Given a hypertext machine, the *hyperconnectivity* of degree n is defined as the set of length-n sequences of links that a user can follow.

$$\begin{aligned} &hyper(n, S_i) \\ &\quad = \{\langle l_1, l_2, \ldots, l_n\rangle \mid S_i \mapsto^{l_1} S_1 \mapsto^{l_2} \ldots \mapsto^{l_n} S_n\} \end{aligned}$$

For a traditional document, i.e. a hyperdocument without any links, the degree-n hyperconnectivity is empty, for any $n > 0$. In the next Section, we show how the notion of hyperconnectivity allows us to compare different hypertext machines.

The *hyperconnectivity* of a hypertext machine is the set of all finite sequences of links that a user can traverse:

$$hyper(S_i) = \bigcup_n hyper(n, S_i)$$

Let us note that, even though links are regarded as primordial in hypertexts, documents cannot be neglected as they contain information. Indeed, it would be unrealistic to consider two hypertext systems as equivalent if they did not contain the same documents. Our notion of hyperconnectivity takes documents into account, because each link specifies the names of a source and a destination document.

2.2. Properties

A hypertext machine $\mathcal{H}_s\langle D_i, LB, \rho, D^*\rangle$ as described in Figure 1 is in fact a regular automaton composed of states in D^*, with an initial state D_i, and transitions between states as defined by links in LB. If all states are defined as accepting states, then hyperconnectivity is the set of words, i.e. the language, accepted by the automaton.

In the sequel, we shall study other forms of hypertext machines, which constitute other types of automata, such as pushdown automata or Turing machines. Therefore, we shall qualify a hypertext machine by it associated automaton, which we also call its *class*.

Definition 2. A hypertext machine is said to be *regular* if it is a non-deterministic finite state automaton. □

As the hypertext machine defined in Figure 1 is regular, the hyperconnectivity of a regular hypertext can be defined by a regular expression (Thm 2.5.1, [25]). Park [14] has proved that the hyperconnectivity of such a hypertext machine is regular. Furuta and Stotts [11] show that such hypertext systems can also be expressed as Petri Nets, and can be described by their Trellis model.

As the hyperconnectivity of a hypertext machine is in fact the language accepted by the machine regarded as an automaton, equality of hypertext machines is derived from the equality of automata. As hyperconnectivity is the property according to which a hypertext system exhibits links between documents, what we call its *expressiveness*, we say that two hypertext machines have the same expressiveness if they have the same degree-n hyperconnectivity, for all n.

Definition 3 (Expressiveness). Two hypertext machines S_1 and S_2 are *as expressive* if for any $n \geq 0$, $hyper(n, S_1) = hyper(n, S_2)$. □

Our first proposition is that valid linkbases increase the hyperconnectivity of hypertext machines when their size increases.

Proposition 2.1. *If a valid linkbase is increasing, so is the hyperconnectivity.*

Proof. Let D^* be a set of documents, and let LB_1, LB_2 be two valid linkbases such that $LB_1 \subseteq LB_2$. Let the states S_1, S_2 be $\mathcal{H}_s\langle D_i, LB_1, \rho, D^*\rangle$, $\mathcal{H}_s\langle D_i, LB_2, \rho, D^*\rangle$, respectively, with $D_i \in D^*$. Then, we can derive that for all n, $hyper(n, S_1) \subseteq hyper(n, S_2)$, as any link l used in a transition for S_1 can also be used for S_2. □

This proposition formally states that links may increase the possibility of navigating in the hyperspace. As a consequence, if adding a link to a hypertext system S strictly increases its hyperconnectivity (strict inclusion $\subset$), then the resulting hypertext has a *different expressiveness* from S.

At this stage, one might wonder whether adding a link increases the expressiveness of the hypertext system. Without doubt, a link offers new navigation possibilities, but it is a user's (or an author's) issue to determine whether the link is useful or adds some meaning. Therefore, we can only conclude that expressiveness changes; we shall come back to this issue later in the paper.

2.3. Other Kinds of Anchors

We regard anchors as the mechanism for referring to a part of a document [9]; unidirectional links are defined by a source anchor and a destination anchor. In Section 2.1, anchors are defined as pairs composed of a document name and an offset; as a result, links are said to be *point to point*.

Point to document links are links whose destination anchor does not specify a document offset. When a point-to-document link is traversed, the document specified by the destination anchor becomes the current focus. Implementations of hypertext systems typically display the destination document at its beginning.

Symmetrically, a *document to point* link is a link whose source anchor specifies a document, but not a document offset. A document-to-point link can be traversed if the document in focus is the document specified by the source anchor.

Finally, a *document to document* link does not specify an offset for the source or destination anchors.

In order to take into account these new kinds of anchors, we extend the hypertext machine as follows:

$$\begin{array}{lcl} \mathit{Source} & = & \mathit{Name} \times \mathit{Offset} + \mathit{Name} \\ & & \qquad \text{(Source Anchor)} \\ \mathit{Dest} & = & \mathit{Name} \times \mathit{Offset} + \mathit{Name} \\ & & \qquad \text{(Destination Anchor)} \end{array}$$

Anchors specify a document name, but are free not to specify an offset. The link traversal relation now supports four kinds of links.

$$\begin{array}{l} \mapsto \subseteq \mathit{State} \times \mathit{State} \times \mathit{Link} \\ \mathcal{H}_{\mathsf{s}}\langle D_1, LB, \rho, D^* \rangle \mapsto^l \mathcal{H}_{\mathsf{s}}\langle D_2, LB, \rho, D^* \rangle \\ \quad \text{if} \quad l = \langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle\rangle,\ l = \langle N_1, \langle N_2, o_2\rangle\rangle, \\ \qquad l = \langle\langle N_1, o_1\rangle, N_2\rangle,\ \text{ or } l = \langle N_1, N_2\rangle \text{ such that} \\ \qquad\quad l \in LB, \\ \qquad\quad \rho(N_1) = D_1,\ \rho(N_2) = D_2, \\ \qquad\quad o_1 \in DOM(D_1), o_2 \in DOM(D_2), \\ \qquad\quad D_1, D_2 \in D^*. \end{array}$$

In practice, hypertext systems with point-to-point links are able to conveniently position the cursor into documents; this feature facilitates and accelerates the access to information. However, our modelling of hypertext systems consider that the current focus is a document as a whole, independently of the current subpart being displayed. As a result, accessing a document via a point-to-document link instead of a point-to-point link does not change the set of links that are available in the destination document.

We can show that the hyperconnectivity generated by point-to-document links is equal to the one generated by point-to-point links (up to a translation), which entails that they contribute to the same hypertext expressiveness. (The same property also holds for document-to-point and document-to-document links.)

Proposition 2.2. *Point-to-document links and point-to-point links generate hyperconnectivities that are equal up to a translation.*

Proof. We can translate a hypertext with point-to-point links into a hypertext with point-to-document links, by translating every point-to-point link $l = \langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle\rangle$ into a point-to-document link $l' = \langle\langle N_1, o_1\rangle, N_2\rangle$. Therefore, a base of point-to-point links LB can be translated into a base of point-to-document links LB'.

As a result, every transition based on point-to-point links

$$\begin{array}{l} \mathcal{H}_{\mathsf{s}}\langle D_1, LB, \rho, D^* \rangle \mapsto^l \mathcal{H}_{\mathsf{s}}\langle D_2, LB, \rho, D^* \rangle \\ \quad \text{if} \quad l = \langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle\rangle \in LB, \end{array}$$

can be mapped to a transition using point-to-document links:

$$\begin{array}{l} \mathcal{H}_{\mathsf{s}}\langle D_1, LB', \rho, D^* \rangle \mapsto^{l'} \mathcal{H}_{\mathsf{s}}\langle D_2, LB', \rho, D^* \rangle \\ \quad \text{if} \quad l' = \langle\langle N_1, o_1\rangle, N_2\rangle \in LB', \end{array}$$

where the destination offset is not specified. Available links are only dependent on the document in focus and not on the current cursor position. Vice-versa, if there is a transition $\mapsto^{l'}$ with $l' \in LB'$, then there is a point-to-point link $l \in LB$, whose translation is l'.

Let us consider a hypertext with point-to-point links S and its translation into a hypertext with point-to-document S'. The systems S and S' are bisimilar because there exists a transition $S \mapsto^l S_1$ if and only if there exists $S' \mapsto^{l'} S'_1$, and S_1 and S'_1 are also bisimilar. Therefore, hyperconnectivities are equal up to the translation of links. □

As point-to-point and point-to-document links generate hyperconnectivities that are equal up to a translation, we can say that they contribute to the same expressiveness, and henceforth the same hypertext class.

Let us note that Proposition 2.2 holds for hypertext machines such as described in Figure 1. SGML [22] and Hytime [26] further introduce two ideas. First, tokens may be interpreted, whereas in our approach they are constant. Second, a presentation layer may also process tokens. Therefore, in the latter systems, retrieving a document may have a different semantics from retrieving the same document at a given offset.

So far, we have only considered unidirectional links, composed of a source anchor and a destination anchor that indicate in which direction a link can be traversed. *Bidirectional* links are also defined by two anchors, but they do not specify the direction of use; in other words, bidirectional links may be traversed in both directions. In our abstract hypertext system, a bidirectional link can be defined in terms of two unidirectional links, with swapped anchors. This definition should be contrasted with users' experience (and authors'), who see a bidirectional link as a single entity, which can be traversed, created, edited, deleted by appropriate actions; it is the role of a hypertext system to map the semantic definition into this single physical reality.

Other kinds of links can also be considered. For example, systems such as Hytime [26] or XLink [27] support *n-ary links*, which involve $n > 2$ anchors [28].

3. EMBEDDED LINKS AND SEPARATE LINKS

The hypertext machine in Section 2 uses a notion of point-to-point link, which we also call *simple link*. So far, we have considered that links are stored in linkbases, which are entities distinct from documents. In other words, links are kept *separate* from documents [29]. In this section, we study another approach where

$$
\begin{array}{lcll}
o \in \mathit{Offset} & = & \{m \in \mathbb{N}, m \geq 0\} & \text{(Offset)} \\
t \in \mathit{Token} & = & \{t_1, t_2, \ldots\} & \text{(Token)} \\
N \in \mathit{Name} & = & \{N_1, N_2, \ldots\} & \text{(Document Name)} \\
\\
A \in \mathit{ADoc} & = & \mathit{Offset} \rightarrow_f \mathit{AToken} & \text{(Annotated Document)} \\
\rho \in \mathit{Env} & = & \mathit{Name} \rightarrow_f \mathit{Doc} & \text{(Naming Function)} \\
A^* \in \mathit{ADocSet} & = & \mathbb{F}(\mathit{ADoc}) & \text{(Set of Annotated Documents)} \\
d^* \in \mathit{AnchorSet} & = & \mathbb{F}(\mathit{Dest}) & \text{(Set of Destination Anchors)} \\
\\
u \in \mathit{AToken} & = & \mathit{Token} \times \mathit{AnchorSet} & \text{(Annotated Token)} \\
d \in \mathit{Dest} & = & \mathit{Name} \times \mathit{Offset} & \text{(Destination Anchor)} \\
S \in \mathit{State} & = & \mathit{ADoc} \times \mathit{Env} \times \mathit{ADocSet} & \text{(State)}
\end{array}
$$

Notation:

$$
\begin{array}{lcll}
u & ::= & \langle t, d^* \rangle & \text{(Annotated Token)} \\
S & ::= & \mathcal{H}_\mathsf{e}\langle A, \rho, A^* \rangle & \text{(State)}
\end{array}
$$

Link traversal relation:

$$
\begin{array}{l}
\mapsto \ \subseteq \ \mathit{State} \times \mathit{State} \times \mathit{Link} \\
\mathcal{H}_\mathsf{e}\langle A_1, \rho, A^* \rangle \mapsto^l \mathcal{H}_\mathsf{e}\langle A_2, \rho, A^* \rangle \\
\quad \text{if} \quad l = \langle\langle N_1, o_1 \rangle, \langle N_2, o_2 \rangle\rangle, \text{ such that } \begin{array}[t]{l} \mathit{valid}(l, \rho, A^*) \\ \rho(N_1) = A_1, \\ \rho(N_2) = A_2. \end{array}
\end{array}
$$

FIGURE 2. State Space and Transition Relation of a Hypertext Machine with Embedded Links

simple links are not stored in linkbases but are embedded explicitly into the documents themselves. In the latter approach, links are said to be *embedded* into documents.

In Figure 2, we redefine the hypertext machine so that it can deal with embedded links. We introduce a new category of *annotated documents*, which map offsets to tokens and a finite set of destination anchors (possibly empty). Destination anchors are meant to be the destination part of a link, with an implicit source anchor, formed of the current document name and the current offset. A state is now a pair composed of the document in focus and a set of annotated documents; the latter containing the former.

In Figure 2, the link traversal relation is adapted accordingly: a transition is allowed between two states S_1, S_2 of a hypertext machine with embedded links if there is an offset in the document in focus that contains an anchor pointing at the document in focus in S_2. The definition of hyperconnectivity is the same as in the previous section.

The approach based on embedded links can be found, for instance, in HTML [6], the markup language adopted by the World Wide Web (WWW) [30]. In HTML, links appear as markups in the document, but HTML allows the user to create one link only at a given offset of a document.

The question of expressiveness of hypertext machines with separate or embedded links can be solved within our proposed framework. Indeed, one can define a "compilation function" that generates a set of annotated documents from a set of documents and a linkbase.

PROPOSITION 3.1. *For a given set of documents and simple links, a hypertext machine with separate links is as expressive as a hypertext system with machine links.*

Proof. Using a compilation function, we can prove that the hyperconnectivity of a hypertext with separate links is equal to the hyperconnectivity of the derived hypertext with embedded links. We shall assume here that we deal with valid links.

We define $\mathcal{E}$ the embedding function transforming a hypertext with separate links into a hypertext with embedded links. It uses an auxiliary function $\mathcal{D}$ translating documents into their annotated version with embedded links.

$$
\begin{array}{l}
\mathcal{E}[\![\mathcal{H}_\mathsf{s}\langle D_1, LB, \rho, D^* \rangle]\!] = \mathcal{H}_\mathsf{e}\langle A_1, \rho', A^* \rangle \\
\quad \text{with} \quad A_i = \mathcal{D}[\![D_i, \rho, LB]\!], \\
\qquad\qquad A^* = \{A_i \mid A_i = \mathcal{D}[\![D_i, \rho, LB]\!], D_i \in D^*\} \\
\qquad\qquad \rho'(N_i) = A_i \text{ iff } \rho(N_i) = D_i, \forall N_i \in \mathit{DOM}(\rho_i)
\end{array}
$$

$$
\begin{array}{l}
\mathcal{D}[\![D, \rho, LB]\!] \\
\quad = \quad \{(o_s, u), \text{such that } u = \langle t, d^* \rangle \\
\qquad\quad \text{with } t = D(o_s)
\end{array}
$$

$$\text{and } d^* = \{\langle N_d, o_d\rangle, \text{ such that } \langle\langle N_s, o_s\rangle, \langle N_d, o_d\rangle\rangle \in LB \text{ and } \rho(N_s) = D\} \}$$

Now, we can easily derive that if $S_1 = \mathcal{H}_\mathsf{s}\langle D_1, LB, \rho, D^*\rangle \mapsto^l S_2 = \mathcal{H}_\mathsf{s}\langle D_2, LB, \rho, D^*\rangle$, then $\mathcal{E}[\![S_1]\!] = \mathcal{H}_\mathsf{e}\langle A_1, \rho', A^*\rangle \mapsto^l \mathcal{E}[\![S_2]\!] = \mathcal{H}_\mathsf{e}\langle A_2, \rho', A^*\rangle$, with $l = \langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle\rangle$.

The second step of the proof is to establish that there is a "decompilation" function $\mathcal{E}^{-1}$, the inverse of $\mathcal{E}$, which can generate a document and a linkbase from an annotated document. We also use an auxiliary function $\mathcal{D}^{-1}$ to generate a document from an annotated document.

$$\begin{aligned}
&\mathcal{E}^{-1}[\![\mathcal{H}_\mathsf{e}\langle A_1, \rho', A^*\rangle]\!] = \mathcal{H}_\mathsf{s}\langle D_1, LB, \rho, D^*\rangle \\
&\quad\text{with} \quad \rho(N_i) = D_i \text{ iff } \rho'(N_i) = A_i, \forall N_i \in DOM(\rho'_i) \\
&\qquad D_i = \mathcal{D}^{-1}[\![A_i]\!] = \{(o_s, t), \text{ such that there exists } d^*, \langle o_s, \langle t, d^*\rangle\rangle \in A_i\} \\
&\qquad LB = \{\ \langle\langle N_i, o_j\rangle, \langle N_k, o_l\rangle\rangle \text{such that } \rho(N_i) = A_i, A_i(o_j) = \langle t, d^*\rangle, \langle N_k, o_l\rangle \in d^*\}
\end{aligned}$$

Now, we can easily derive that if $S_1 = \mathcal{H}_\mathsf{e}\langle A_1, \rho, A^*\rangle \mapsto^l S_2 = \mathcal{H}_\mathsf{e}\langle A_2, \rho, A^*\rangle$, then $\mathcal{E}^{-1}[\![S_1]\!] = \mathcal{H}_\mathsf{s}\langle D_1, LB, \rho', D^*\rangle \mapsto^l \mathcal{E}^{-1}[\![S_2]\!] = \mathcal{H}_\mathsf{s}\langle D_2, LB, \rho', D^*\rangle$, with $l = \langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle\rangle$.

Furthermore, the composition of translations $\mathcal{E}[\![\mathcal{E}^{-1}[\![S]\!]]\!]$ and $\mathcal{E}^{-1}[\![\mathcal{E}[\![S]\!]]\!]$ is the identity function. As a result, the hyperconnectivities are equal. □

The compilation function $\mathcal{E}$ and its inverse $\mathcal{E}^{-1}$ are in fact the formalisation of Brown and Brown's [8] *Joiner* and *Splitter* software components, respectively. They conceived such converters for practical reasons: the joiner and the splitter are a practical means to offer the best of both approaches, embedded vs. separate links. On the other hand, we derived the compilation functions $\mathcal{E}$ and $\mathcal{E}^{-1}$ in order to prove that hyperconnectivities of both types of hypertext systems are the same.

Even though they have the same expressiveness as embedded links, separate links can offer more flexibility. *(i)* Linkbases are extensible without recompilation. *(ii)* Several linkbases can be used for a given set of documents, according to the user's needs. On the other hand, separate links also have disadvantages; in particular, the *editing problem* [17, 31] is concerned with maintaining the integrity of links when editing a document with separate links.

4. GENERIC LINKS

Let us again consider hypertext systems with separate links. In Section 2, we stated that *simple links* associate a source and a destination anchor. Some hypertext systems allow the source anchor to be dynamic; that is, even though the source anchor is specified at authoring-time, it is not actually manifested before navigation time. In particular, Microcosm [16, 17] introduces the notion of *generic link* able to connect any occurrence of a keyword in any document to a particular destination anchor. A benefit of such generic links is that they require much lower authoring effort, which we shall explain later.

In Figure 3, we extend the state space of Figure 1 with generic links. Linkbases can contain either simple links or generic links. A generic link is a pair formed of a token and a destination anchor. Such a generic link means that for every such token appearing in the set of documents, there is a conceptual link to the destination anchor that can be traversed. In Figure 3, the link traversal relation is changed accordingly: users may traverse simple or generic links.

Given a set of documents, generic links may be "compiled" into a linkbase of simple links.

Proposition 4.1. *For a fixed hypertext with generic links and a given set of documents, there exists a hypertext with simple links that is as expressive.*

Proof. We define $\mathcal{S}$, a compilation function, which generates a base of simple links for a given set of generic links and a given set of documents. The compilation function $\mathcal{S}$ is recursive. In the base case, we deal with an empty set of generic links. In the inductive case, we deal with a non-empty set $\{g\} \cup G$ of generic links, containing a generic link g, the rest of the set being denoted by G.

$$\begin{aligned}
\mathcal{S}[\![\emptyset, \rho, D^*]\!] &= \emptyset \\
\mathcal{S}[\![\{g\} \cup G, \rho, D^*]\!] &= \mathcal{S}[\![G, \rho, D^*]\!] \cup LB \ \text{ with} \\
LB = \{l = \langle\langle N_1, o_1\rangle, &\langle N_2, o_2\rangle\rangle \mid g = \langle t, \langle N_2, o_2\rangle\rangle, \\
&\rho(N_1) = D_1, D_1(o_1) = t, valid(l, \rho, D^*)\}
\end{aligned}$$

As a result $\mathcal{H}_\mathsf{s}\langle D_1, LB, \rho, D^*\rangle \mapsto^l \mathcal{H}_\mathsf{s}\langle D_2, LB, \rho, D^*\rangle$ if and only if $\mathcal{H}_\mathsf{g}\langle D_1, G, \rho, D^*\rangle \mapsto^l \mathcal{H}_\mathsf{s}\langle D_2, G, \rho, D^*\rangle$, when $LB = \mathcal{S}[\![G, \rho, D^*]\!]$.

Let us observe that the transition arrow $\mapsto^l$ is annotated with the concrete source and destinations anchors, i.e. a simple link, when a generic link is followed. Therefore, we can derive that for all n, $hyper(n, \mathcal{H}_\mathsf{s}\langle D_i, LB, \rho, D^*\rangle)$ is equal to $hyper(n, \mathcal{H}_\mathsf{g}\langle D_i, G, \rho, D^*\rangle)$, for any $D_i \in D^*$, if $LB = \mathcal{S}[\![G, \rho, D^*]\!]$. □

From the *reader's viewpoint*, generic links do not change the expressiveness of hypertext systems with simple links, because for a given set of documents, there is always a base of simple links that provide the same hyperconnectivity, i.e. that provide the reader with the same navigation ability.

$$
\begin{array}{lcll}
o \in \mathit{Offset} & = & \{m \in \mathbb{N}, m \geq 0\} & \text{(Offset)}\\
t \in \mathit{Token} & = & \{t_1, t_2, \ldots\} & \text{(Token)}\\
N \in \mathit{Name} & = & \{N_1, N_2, \ldots\} & \text{(Document Name)}\\
\\
D \in \mathit{Doc} & = & \mathit{Offset} \rightarrow_f \mathit{Token} & \text{(Document)}\\
\rho \in \mathit{Env} & = & \mathit{Name} \rightarrow_f \mathit{Doc} & \text{(Naming Function)}\\
LB \in \mathit{LBSet} & = & \mathbb{F}(\mathit{Link} \cup \mathit{GLink}) & \text{(Linkbase)}\\
D^* \in \mathit{DSet} & = & \mathbb{F}(\mathit{Doc}) & \text{(Set of Documents)}\\
\\
l \in \mathit{Link} & = & \mathit{Source} \times \mathit{Dest} & \text{(Link)}\\
s \in \mathit{Source} & = & \mathit{Name} \times \mathit{Offset} & \text{(Source Anchor)}\\
d \in \mathit{Dest} & = & \mathit{Name} \times \mathit{Offset} & \text{(Destination Anchor)}\\
g \in \mathit{GLink} & = & \mathit{Token} \times \mathit{Dest} & \text{(Generic Link)}\\
S \in \mathit{State} & = & \mathit{Doc} \times \mathit{LBSet} \times \mathit{Env} \times \mathit{DSet} & \text{(State)}
\end{array}
$$

Notation:

$$
\begin{array}{lcll}
g & ::= & \langle t, d \rangle & \text{(Generic Link)}\\
S & ::= & \mathcal{H}_{\mathrm{g}}\langle D, LB, \rho, D^* \rangle & \text{(State)}
\end{array}
$$

Link traversal relation:

$$
\begin{array}{l}
\mapsto \ \subseteq \ \mathit{State} \times \mathit{State} \times \mathit{Link}\\
\mathcal{H}_{\mathrm{g}}\langle D_1, LB, \rho, D^* \rangle \mapsto^l \mathcal{H}_{\mathrm{g}}\langle D_2, LB, \rho, D^* \rangle\\
\text{with } l = \langle \langle N_1, o_1 \rangle, \langle N_2, o_2 \rangle \rangle\\
\quad \text{if} \quad \rho(N_1) = D_1,\ \rho(N_2) = D_2, \mathit{valid}(l, \rho, D^*) \text{and} \left\{ \begin{array}{l} \text{either } l \in LB,\\ \text{or } \exists g = \langle t, \langle N_2, o_2 \rangle \rangle \in LB, D_1(o_1) = t. \end{array} \right.
\end{array}
$$

FIGURE 3. State Space and Transition Relation of a Hypertext Machine with Simple and Generic Links

However, generic links are useful because they require less effort to author a hypertext system. Intuitively, an author needs less generic links than simple links in order to achieve a given hyperconnectivity. Let us note that we measure the benefit for the *authors* by comparing the effect of links on the hyperconnectivity seen by the *reader*.

The following proposition requires the notions of *valid* generic link and *simple-link instance*. In this context, validity of a generic link means that there exists a document to which the generic link is applicable, and such that its destination is not dangling.

DEFINITION 4. A generic link $g = \langle t, \langle N_2, o_2 \rangle \rangle$ is *valid* with respect to a a naming function ρ and a set of documents D^*, written $\mathit{valid}(g, D^*, \rho)$, if the following conditions hold, assuming that $D_2 = \rho(N_2)$.

1. there exists N_1, $D_1 = \rho(N_1)$ and $D_1 \in D^*$,
2. there exists $o_1 \in \mathit{DOM}(D_1)$,
3. $D_1(o_1) = t$,
4. $D_2 \in D^*$,
5. $o_2 \in \mathit{DOM}(D_2)$.

□

DEFINITION 5. A linkbase LB only composed of simple links is a *simple-link instance* of a valid set G of generic links with respect to a naming function ρ and a set of documents D^*, if LB is a set such that there is a one to one relation with G: $\forall g \in G$, $g = \langle t, \langle N_2, o_2 \rangle \rangle$, then $\exists l \in LB$, such that $l = \langle \langle N_1, o_1 \rangle, \langle N_2, o_2 \rangle \rangle$, with $D_1 = \rho(N_1), D_1(o_1) = t$ and $D_1, D_2 \in D^*$. □

The following proposition states that a base of generic links creates a greater or equal hyperconnectivity than any of its simple-link instances.

PROPOSITION 4.2. *Let LB be a simple-link instance of a set G of valid generic links, with respect to a set of documents D^*, then for all n, $\mathit{hyper}(n, \mathcal{H}_{\mathrm{g}}\langle D, G, \rho, D^* \rangle) \supseteq \mathit{hyper}(n, \mathcal{H}_{\mathrm{s}}\langle D, LB, \rho, D^* \rangle)$, for any $D \in D^*$.*

Proof. We have that $\mathit{hyper}(n, \mathcal{H}_{\mathrm{g}}\langle D, G, \rho, D^* \rangle) = \mathit{hyper}(n, \mathcal{H}_{\mathrm{s}}\langle D, LB, \rho, D^* \rangle)$ if $\forall g \in G$, $g = \langle t, \langle D_2, o_2 \rangle \rangle$, there exists one and only document $D' \in D^*$ and one and only one offset o_1 such that $D'(o_1) = t$. Otherwise, let $LB' = \mathcal{S}[\![G, \rho, D^*]\!]$. Then, $\mathit{hyper}(n, \mathcal{H}_{\mathrm{g}}\langle D, G, \rho, D^* \rangle) = \mathit{hyper}(n, \mathcal{H}_{\mathrm{s}}\langle D, LB', \rho, D^* \rangle)$ by Proposition 4.1, and $\mathit{hyper}(n, \mathcal{H}_{\mathrm{s}}\langle D, LB', \rho, D^* \rangle) \supseteq \mathit{hyper}(n, \mathcal{H}_{\mathrm{s}}\langle D, LB, \rho, D^* \rangle)$, as $LB' \supseteq LB$, by Proposition 2.1. □

Generic links are flexible: the set of documents of a hypertext systems may be extended, and generic links will be applicable to the new documents.

Some might argue that generic links are *too powerful* because specifying a generic link is equivalent to blindly creating links for every occurrence of a token in a set of documents, independently of its context or its actual meaning at this offset. Therefore, it is a point of debate that generic links increase the expressiveness of hypertext systems.

Such a discussion introduces the notion of the *semantic validity* of a link. Intuitively, we say that a link is semantically valid if its presence "makes some sense". Even though such a notion of semantic validity is probably not computable, we can use algorithmic methods to approximate such a criterion. Microcosm [16, 17] introduces *local links* as a kind of generic link restricted to a scope defined by a set of documents; such local links can be defined straightforwardly in our abstract model, by adding a set of documents restricting the scope of a generic link.

In Section 7, we shall present a mechanism that provides an automated dynamic linking mechanism that generalises generic links by allowing computation that can be used to specify semantically correct networks of links.

5. HISTORY MECHANISM

Numerous hypertext systems (WWW browsers, Intermedia [7], Microcosm [16, 17]) maintain the history of followed links and/or of accessed documents. In addition, some of them provide the user with the ability to step backwards in the navigation. Typically, they offer a "back" button, which restores the latest navigation state in the history, i.e. the latest document and offset. In Figure 4, we extend our hypertext machine with a history mechanism. The history is defined as a list of links. The empty list is noted [], whereas the non-empty list $[l : H]$ is composed of a head l and a tail H. The history is added as a new explicit component of the machine state. As each simple link contains a source and a destination anchor, the history not only contains the links that were followed, but also the names of the documents that were accessed.

The link traversal relation is adapted accordingly. Forward navigation, noted $\mapsto^l$, is defined as the action of following a link l, which results in adding it to the history of the state. Backward navigation is the ability to follow the latest link that appears in the history in reverse direction: the hypertext system is back to the state that existed before this link was followed forward. Such a backward navigation is noted $\mapsto^b$, where the symbol b indicates that we are reversing the last link traversed. The transition b is permitted if the history list is not empty, which implies that users cannot go backwards more than they go forward.

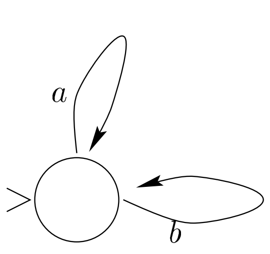
Bieber and Wan [33] study the problem of backtracking in a multiple-window environment: there, instead of having a single document in focus, each window contains a document in focus. They distinguish chronological from task-based backtracking; they also discuss the issue of whether previous states should be deleted from the history list after backtracking.

Let us note that the history mechanism described here is a formalisation of the notion of trails [34].

A history mechanism differs from bidirectional links. With the former, hypertext systems become a form of pushdown automaton and have a similar expressiveness as context-free grammars [25]. On the other hand, hypertext systems with bidirectional links are still regular: the bidirectionality is a property of the links in the linkbase.

Proposition 5.1. A hypertext system with history mechanism and a back button has a different expressiveness from a hypertext system with bidirectional links.

Proof. Let us consider a hypertext machine with a single document, represented as a node in the following picture. We add to this hypertext system a bidirectional link (from the document to itself); the forward direction is represented by the simple link a and the backward direction by the simple link b. Such a hypertext with bidirectional links is represented below; the circle represents the document in focus, the arrows are links that can be followed, and the > sign is the initial state of the system.



As the user has the possibility to follow the a or the b links, the degree-n hyperconnectivity can be described by word of length n generated by the regular expression: $\{a, b\}^*$.

Now, let us consider a hypertext system with the same document and the same unidirectional link a. With a history mechanism and a back button, whose transitions are represented by b, such a hypertext system generates link sequences of the type

$$(a^{n_i} b^{m_i})^*, \text{ with } \sum_{j \le i} m_j \le \sum_{j \le i} n_j.$$

This expression ensures that a user can always follow the forward link a. The inequality ensures that the total number of backward transitions is bounded by the total number of forward transitions.

In order to prove that the hypertext has indeed such a hyperconnectivity, one can observe that the length of the history list is given by

$$\sum_{j \le i} n_j - \sum_{j \le i} m_j.$$

$$\begin{array}{lcll}
o \in \mathit{Offset} & = & \{m \in \mathbb{N}, m \geq 0\} & \text{(Offset)} \\
t \in \mathit{Token} & = & \{t_1, t_2, \ldots\} & \text{(Token)} \\
N \in \mathit{Name} & = & \{N_1, N_2, \ldots\} & \text{(Document Name)} \\
\\
D \in \mathit{Doc} & = & \mathit{Offset} \rightarrow_f \mathit{Token} & \text{(Document)} \\
\rho \in \mathit{Env} & = & \mathit{Name} \rightarrow_f \mathit{Doc} & \text{(Naming Function)} \\
LB \in \mathit{LBSet} & = & \mathbb{F}(\mathit{Link}) & \text{(LinkBase)} \\
D^* \in \mathit{DSet} & = & \mathbb{F}(\mathit{Doc}) & \text{(Set of Documents)} \\
\\
l \in \mathit{Link} & = & \mathit{Source} \times \mathit{Dest} & \text{(Link)} \\
s \in \mathit{Source} & = & \mathit{Name} \times \mathit{Offset} & \text{(Source Anchor)} \\
d \in \mathit{Dest} & = & \mathit{Name} \times \mathit{Offset} & \text{(Destination Anchor)} \\
H \in \mathit{Hist} & = & \{[]\} \ + \ \mathit{Link} \times \mathit{Hist} & \text{(History)} \\
\\
S \in \mathit{State} & = & \mathit{Doc} \times \mathit{LBSet} \times \mathit{Env} \times \mathit{Hist} & \text{(State)}
\end{array}$$

Notation:

$$\begin{array}{lcll}
H & ::= & [] \mid [l : H] & \text{(History)} \\
S & ::= & \mathcal{H}_{\mathsf{h}}\langle D, LB, \rho, D^*, H\rangle & \text{(State)}
\end{array}$$

$$\begin{array}{l}
\mapsto \ \subseteq \ \mathit{State} \times \mathit{State} \times (\mathit{Link} \cup \{b\}) \\
\quad \mathcal{H}_{\mathsf{h}}\langle D_1, LB, \rho, D^*, H\rangle \mapsto^l \mathcal{H}_{\mathsf{h}}\langle D_2, LB, \rho, D^*, [l : H]\rangle \\
\qquad \text{if} \quad l = \langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle\rangle, \mathit{valid}(l, \rho, D^*) \\
\quad \mathcal{H}_{\mathsf{h}}\langle D_1, LB, \rho, D^*, [l : H]\rangle \mapsto^b \mathcal{H}_{\mathsf{h}}\langle D_2, LB, \rho, D^*, H\rangle \\
\qquad \text{if} \quad l = \langle\langle N_2, o_2\rangle, \langle N_1, o_1\rangle\rangle, \mathit{valid}(l, \rho, D^*), \rho(N_1) = D_1, \rho(N_2) = D_2
\end{array}$$

FIGURE 4. State Space and Transition Relation of a Hypertext Machine with History

Therefore, following [25], the latter language is not regular, but it can be recognised by a pushdown automaton. □

Proposition 5.1 introduces a new kind of hypertext systems, which we call *pushdown* [25].

DEFINITION 6. A *pushdown* hypertext machine is an abstract machine that behaves as a pushdown automaton. □

In practical terms, Proposition 5.1 states that the history mechanism and the back button introduce a new class of hypertext systems, whose navigation space, i.e. hyperconnectivity, can only be provided by a more powerful class of hypertext systems.

It should be observed that the hyperconnectivity generated by links is in fact specified by the authors when they author documents. The hyperconnectivity derived from back buttons offers new navigation capabilities to the user, but it is provided by the hypertext system itself. In the following sections, we present other mechanisms of linking that allow authors to specify the hyperconnectivity they wish.

6. ADAPTIVE LINKING

Let us consider a hypertext system composed of three documents, namely a root document R, a definition document D, and a properties document P, with D and P accessible from R. Figure 5 is a graphical representation of such a system, where states S_R, S_D, S_P have R, D and P in focus, respectively. The > sign on S_R indicates that it is the initial state.

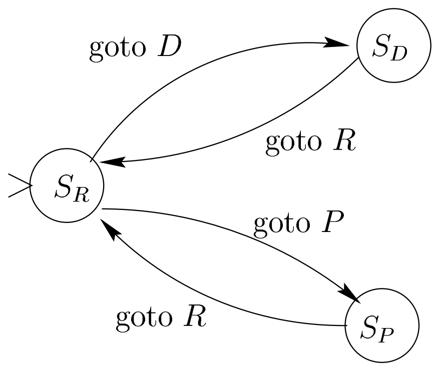


FIGURE 5. A Regular Hypertext

Now, let us imagine that we want to build a hypertext system where the properties document can only be accessed after the definition document has been browsed. Figure 6 models such a system: a graph node represents not only the document that is currently displayed, but also its associated "state". For instance, in S_R and

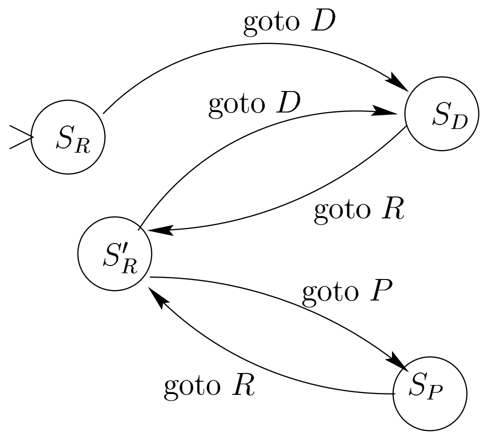


FIGURE 6. Adaptive Linking

S'_R, the root document is in focus, but S'_R carries the additional knowledge that the definition document has already been accessed.

Hypertext systems as in Figure 6 are said to have *adaptive links* because the links that are available from a given document are dependent on the navigation history. As a matter of fact, such a hypertext can be modelled by extending the hypertext machine with *several linkbases*. Adaptive linking is a special case of adaptive hypertexts, where links and document contents may be dependent on the navigation history and on the user model. In this section, we shall only focus on a specific facet of adaptive hypertexts: we describe a mechanism by which links may be adapted according to the navigation.

In the state space of Figure 7, we distinguish two types of links: simple links of the form $\langle s, d, - \rangle$ specify source and destination anchors, whereas "*linkbase changing*" links $\langle s, d, N^{LB} \rangle$ further specify the name of a linkbase that must become active after being followed. Such a linkbase changing link is similar to Deschamps' content changing links [35].

The link traversal relation distinguishes the two kinds of links. As previously, following a simple link changes the document in focus and leaves the current linkbase untouched. The novelty is that traversing a linkbase-changing link sets a new linkbase in the machine state for subsequent navigation.

The two hypertext machines of Figures 5 and 6 have a different expressiveness because their hyperconnectivities are different. Indeed, two transitions are allowed from S_R in Figure 5, whereas only one is legal from S_R in Figure 6. However, we can show that hypertext systems with adaptive linking as described above are still regular.

Proposition 6.1. Hypertext systems with adaptive linking are regular.

Proof. Let n be the number of linkbases in LB^* and m the number of documents in D^*. Let us consider their names $\{N_1, \ldots, N_m\}$. Then, we can define a regular automaton composed of $n \times m$ nodes, which can be enumerated as follows: $\{N_1^1, \ldots, N_m^1, \ldots, N_1^n, \ldots, N_m^n\}$. There is a link between N_i^k and N_j^k if there is a link $\langle N_i, N_j, - \rangle$ in LB_k. There is a link between N_i^k and N_j^l if there is a link $\langle N_i, N_j, N_l^{LB} \rangle$ in LB_k, with $\rho(N_l^{LB}) = LB_l$.

Hence, it is straightforward to prove that *(i)* the derived automaton is regular, *(ii)* its hyperconnectivity is equal to the hyperconnectivity of the initial automaton, up to the translation of links. □

Other adaptive behaviours for hypertexts have been defined. Stotts and Furuta [12] describe dynamic adaptations of the hypertext structure based on time. Their approach supports help menus to appear after a (relative) amount of time. They are also able to make links invisible by delaying their usability indefinitely. The adaptive hypertext that we describe can partly simulate the functionalities offered by Stotts and Furuta's time-based adaption; however, the latter is more powerful, especially by including a notion of time, absent from our model. Link popularity counts [36] were introduced to help the user to navigate in hypertexts; such counts could also be used to determine which links are active.

So far, our survey of linking mechanisms has shown that all linking mechanisms generate regular hyperconnectivities. The back button only, combined with the history list, changes the navigation space because the hypertext system behaves as a pushdown automaton. In the following section, we introduce a more powerful linking mechanism.

7. COMPUTED ANCHORS

Microcosm generic links [16, 17] associate a destination anchor with source anchors that designate a given token. One can generalise these generic links by accepting the source anchor to be computed by an arbitrary function. We can even further generalise the latter approach by allowing the destination anchor to be computed also by a function. We obtain a general notion of link, called the *functional link* by Ashman and Verbyla [18, 19, 37]. Such a functional link is a powerful concept because both its source and destination anchors can be computed by arbitrary functions at navigation time, but also these functions can take decision based on the navigation history. In Figure 8, we present a variant of the functional link, adapted to our hypertext machine.

We see that a link is defined by a pair of functions, respectively called *functional source anchor* and *functional destination anchor*. The former is a predicate, which, for a document name, a document, an offset, and a history list, indicates whether the anchor formed of the document and offset is the source of a link. The latter is a relation, which, for a document name, a doc-

$$\begin{array}{lcll}
o \in \mathit{Offset} & = & \{m \in \mathbb{N}, m \geq 0\} & \text{(Offset)} \\
t \in \mathit{Token} & = & \{t_1, t_2, \ldots\} & \text{(Token)} \\
N \in \mathit{Name} & = & \{N_1, N_2, \ldots\} & \text{(Document Name)} \\
N^{LB} \in \mathit{LBName} & = & \{N_1^{LB}, N_2^{LB}, \ldots, N_m^{LB}\} & \text{(Link Base Name)} \\
s \in \mathit{LBSpec} & = & \mathit{LBName} + \{-\} & \text{(Link Base Spec)} \\
\\
D \in \mathit{Doc} & = & \mathit{Offset} \rightarrow_f \mathit{Token} & \text{(Document)} \\
\rho \in \mathit{Env} & = & (\mathit{Name} \rightarrow_f \mathit{Doc}) + (\mathit{LBName} \rightarrow_f \mathit{LBSet}) & \text{(Naming Function)} \\
LB \in \mathit{LBSet} & = & \mathbb{F}(\mathit{Link}) & \text{(LinkBase)} \\
D^* \in \mathit{DSet} & = & \mathbb{F}(\mathit{Doc}) & \text{(Set of Documents)} \\
LB^* \in \mathit{LBSet}^* & = & \mathbb{F}(\mathbb{F}(\mathit{Link})) & \text{(Set of LinkBases)} \\
\\
l \in \mathit{Link} & = & \mathit{Source} \times \mathit{Dest} \times \mathit{LBSpec} & \text{(Link)} \\
s \in \mathit{Source} & = & \mathit{Name} \times \mathit{Offset} & \text{(Source Anchor)} \\
d \in \mathit{Dest} & = & \mathit{Name} \times \mathit{Offset} & \text{(Destination Anchor)} \\
S \in \mathit{State} & = & \mathit{Doc} \times \mathit{LBSet} \times \mathit{Env} \times \mathit{DSet} \times \mathit{LBSet}^* & \text{(State)}
\end{array}$$

Notation:

$$\begin{array}{lcll}
l & ::= & \langle s, d, -\rangle \mid \langle s, d, N^{LB}\rangle & \text{(Link)} \\
S & ::= & \mathcal{H}_{\mathsf{a}}\langle D, LB, \rho, D^*, LB^*\rangle & \text{(State)}
\end{array}$$

Link traversal relation:

$$\begin{array}{l}
\mapsto \;\subseteq\; \mathit{State} \times \mathit{State} \times \mathit{Link} \\
\mathcal{H}_{\mathsf{a}}\langle D_1, LB, \rho, D^*, LB^*\rangle \mapsto^l \mathcal{H}_{\mathsf{a}}\langle D_2, LB, \rho, D^*, LB^*\rangle \\
\quad \text{if } \; l = \langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle, -\rangle, \text{ such that} \\
\qquad D_1 = \rho(N_1), D_2 = \rho(N_2), l \in LB, \mathit{valid}(\langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle\rangle, \rho, D^*) \\
\mathcal{H}_{\mathsf{a}}\langle D_1, LB_1, \rho, D^*, LB^*\rangle \mapsto^l \mathcal{H}_{\mathsf{a}}\langle D_2, LB_2, \rho, D^*, LB^*\rangle \\
\quad \text{if } \; l = \langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle, N_2^{LB}\rangle, l \in LB_1, \text{ such that} \\
\qquad D_1 = \rho(N_1), D_2 = \rho(N_2), LB_2 = \rho(N_2^{LB}), \;\; LB_1, LB_2 \in LB^*, \text{and } \mathit{valid}(\langle\langle N_1, o_1\rangle, \langle N_2, o_2\rangle\rangle, \rho, D^*).
\end{array}$$

FIGURE 7. State Space and Transition Relation of an Adaptive Hypertext Machine

ument, an offset, and a history list, returns a set of destination anchors. The link traversal relation is now defined in terms of functional anchors in Figure 8.

The next proposition establishes that functional links generalise all the kinds of links previously studied in this paper.

PROPOSITION 7.1. Functional links can express simple, generic, adaptive, or Microcosm local links.

Proof.

1. Let $\langle s, d\rangle$ be a simple link. The corresponding functional link $\langle f_s, f_d\rangle$ can be defined as:

$$\begin{array}{lcl}
f_s & : & \lambda N, D, o, H.\ \langle N, o\rangle = s \\
f_d & : & \lambda N, D, o, H.\ \{d\}
\end{array}$$

2. Let $\langle t, d\rangle$ be a generic link. The corresponding functional link $\langle f_s, f_d\rangle$ can be defined as:

$$\begin{array}{lcl}
f_s & : & \lambda N, D, o, H.\ D(o) = t \\
f_d & : & \lambda N, D, o, H.\ \{d\}
\end{array}$$

3. Using Proposition 6.1, a hypertext with adaptive links may be represented by a regular automaton. Now we can adopt the representation (1) for simple links.
4. In Microcosm, a local link is a kind of generic link $\langle t, d\rangle$, whose scope is restricted to a given set of documents D^*. The corresponding functional link $\langle f_s, f_d\rangle$ can be defined as:

$$\begin{array}{lcl}
f_s & : & \lambda N, D, o, H.\ D(o) = t \ \wedge D \in D^* \\
f_d & : & \lambda N, D, o, H.\ \{d\}
\end{array}$$

□

We can further strengthen this result about the expressiveness of functional links. In the next proposition, we show that a hypertext machine with functional links can emulate any finite computation by a Turing machine [25]. Such a result effectively proves that functional links introduce a new class of hypertext systems. Let us note that we have to restrict ourselves to finite behaviours of Turing machines because our definition of an abstract hypertext system is essentially finite: documents are in finite number and of finite size and linkbases have a finite size.

$$\begin{array}{lcll}
o \in \mathit{Offset} & = & \{m \in \mathbb{N}, m \geq 0\} & \text{(Offset)} \\
t \in \mathit{Token} & = & \{t_1, t_2, \ldots\} & \text{(Token)} \\
N \in \mathit{Name} & = & \{N_1, N_2, \ldots\} & \text{(Document Name)} \\
b \in \mathit{Bool} & = & \{\mathit{true}, \mathit{false}\} & \text{(Boolean)} \\
\\
D \in \mathit{Doc} & = & \mathit{Offset} \rightarrow_f \mathit{Token} & \text{(Document)} \\
\rho \in \mathit{Env} & = & \mathit{Name} \rightarrow_f \mathit{Doc} & \text{(Naming Function)} \\
f_s \in \mathit{FSA} & = & \mathit{Name} \times \mathit{Doc} \times \mathit{Offset} \times \mathit{Hist} \rightarrow \mathit{Bool} & \text{(Functional Source Anchor)} \\
f_d \in \mathit{FDA} & = & \mathit{Name} \times \mathit{Doc} \times \mathit{Offset} \times \mathit{Hist} \rightarrow \mathit{ASet} & \text{(Functional Destination Anchor)} \\
\mathit{LB} \in \mathit{LBSet} & = & \mathbb{F}(\mathit{Link}) & \text{(LinkBase)} \\
D^* \in \mathit{DSet} & = & \mathbb{F}(\mathit{Doc}) & \text{(Set of Documents)} \\
A^* \in \mathit{ASet} & = & \mathbb{F}(\mathit{Anchor}) & \text{(Set of Anchors)} \\
\\
l \in \mathit{Link} & = & \mathit{FSA} \times \mathit{FDA} & \text{(Functional Link)} \\
s, d \in \mathit{Anchor} & = & \mathit{Name} \times \mathit{Offset} & \text{(Anchor)} \\
\ell \in \mathit{LinkInst} & = & \mathit{Anchor} \times \mathit{Anchor} & \text{(Link Instance)} \\
H \in \mathit{Hist} & = & \{[]\} \; + \; \mathit{Link} \times \mathit{Hist} & \text{(History)} \\
\\
S \in \mathit{State} & = & \mathit{Doc} \times \mathit{LBSet} \times \mathit{Env} \times \mathit{DSet} \times \mathit{Hist} & \text{(State)}
\end{array}$$

Notation:

$$\begin{array}{lcll}
l & ::= & \langle f_s, f_d \rangle & \text{(Functional Link)} \\
\ell & ::= & \langle s, d \rangle & \text{(Link Instance)} \\
H & ::= & [] \mid [\ell : H] & \text{(History)} \\
S & ::= & \mathcal{H}_{\mathsf{f}}\langle D, LB, \rho, D^*, H \rangle & \text{(State)}
\end{array}$$

Link traversal relation:

$$\begin{array}{l}
\mapsto \quad \subseteq \quad \mathit{State} \times \mathit{State} \times \mathit{LinkInst} \\
\mathcal{H}_{\mathsf{f}}\langle D_1, LB, \rho, D^*, H \rangle \mapsto^{\ell} \mathcal{H}_{\mathsf{f}}\langle D_2, LB, \rho, D^*, [\ell : H] \rangle \\
\quad \text{with} \quad \ell = \langle \langle N_1, o_1 \rangle, \langle N_2, o_2 \rangle \rangle \quad \text{such that} \quad
\begin{array}[t]{l}
\mathit{valid}(\ell, \rho, D^*) \\
\exists l = \langle f_s, f_d \rangle \in LB, \\
D_1 = \rho(N_1), D_2 = \rho(N_2), \\
f_s(N_1, D_1, o_1, H) \\
\langle N_2, o_2 \rangle \in f_d(N_1, D_1, o_1, H)
\end{array}
\end{array}$$

FIGURE 8. State Space and Transition Relation of a Hypertext Machine with Computed Anchors

PROPOSITION 7.2. A hypertext system with functional links can emulate any finite computation of a Turing machine.

Proof. Let us consider a Turing machine. In order to prove this proposition, we construct a hypertext machine that can emulate the Turing machine. The hypertext machine is composed of a single document. Its content is irrelevant to the proof, but the size of its domain is defined by the biggest value necessary to encode both the value stored on the ribbon of a Turing Machine and the head position [25].

We emulate a Turing machine with our hypertext system as follows. At every step in the navigation, a single link is accessible; the offset of its source anchor is the value encoded in the ribbon of the Turing machine. Initially, a link will be accessible at each offset of the document to specify the initial value of the ribbon. Computation ends when no link can be activated; the final value of the computation is the offset of the last followed link.

We assume that we have a function turing, which for an encoding of the ribbon value and head position in input, gives the encoding of the ribbon value and head position after one transition of the Turing machine. We can define the following functional link $\langle f_s, f_d \rangle$:

$$\begin{array}{lcl}
f_s & = & \lambda N, D, o, H. \\
 & & \quad H = [] \;\text{ or} \\
 & & \quad \text{let } \; [\langle \langle N_1, o_1 \rangle, d \rangle : H'] = H \\
 & & \quad \text{in } \mathsf{turing}(o_1) = o \\
 & & \quad \text{end} \\
f_d & = & \lambda N, D, o, H. \;\; \{\langle N, o \rangle\}
\end{array}$$

The functional anchors specify that there is only one link that is available after $n > 1$ steps, at the offset defined by the ribbon of the Turing machine. Initially, links at every position may be followed to specify the input to the computation. □

Let us note that the proof of Proposition 7.2 involves a somewhat tricky encoding of the state of a Turing machine in the offset of available links. The reason is that the set of observables of our current definition of a hypertext machine is based on links uniquely. However, we can imagine that functional links can modify a dynamic environment and that the latter is made observable: the proof of the proposition will become straightforward, and we might be able to consider infinite computations as well. Some authors have investigated such a direction: e.g, links may be regarded as arbitrary functions or processes [38], or scripts, also called "agents", can be initiated when following links [12]. For instance, Microcosm filters [16, 17] perform some processing when links are followed. Finally, let us observe that the restriction on the finiteness of computations in Proposition 7.2 is due to the fact that documents have a finite size, and not due to a restriction on functional links.

Functional links may be used to define various kinds of other links. Browsers usually indicate which links have been followed by the user, for instance by changing an attribute such as their colour. We can regard this attribute as computed by a functional link.

Let us consider a simple link $\langle s, d\rangle$. We can define a functional link that associates the same source and destination anchors but can only be followed once.

$$\begin{array}{lcl} f_s & : & \lambda N, D, o, H.\ s = \langle N, o\rangle \ \wedge \ \langle s, d\rangle \notin H. \\ f_d & : & \lambda N, D, o, H.\ \{d\} \end{array}$$

Typed links are regarded as key elements in hypertext systems as they enforce the regularity of the structure [39], and they help user's navigating as types represent link meaning. Ashman and Verbyla [19] argue that functional links can encode types as all the links of a given type can be represented by a single functional link.

In Figure 8, our hypertext machine used a history mechanism as in Section 5. In fact, we could accumulate more dynamic information as navigation proceeds. For instance, the current active context [40], attributes of the links followed, etc. could be collected in an *environment*. Functional anchors could take this environment as argument and provide for a more refined form of linking depending on the user's navigation history.

This section has shown us that with functional links, hyperconnectivity is no longer a regular or a context-free language, but is generated by a Turing complete process. In practical terms, this allows authors to specify links that will be manifested at navigation time, depending on the reader's traversal history.

8. DISCUSSION AND RELATED WORK

The goal of this paper is to study linking mechanisms and their expressiveness. There are however numerous hypertext concepts that we would like to formalise. We believe that this hypertext machine approach can be extended to support them. The machine was designed to cope with parallelism and distribution; similar formalisms have been used by the first author to model parallel and distributed programming languages [41]. Hypertext systems with multiple windows can also be modelled by allowing several documents to be in focus. Multiple users and user's models could also been brought into this framework.

The introduction described how Brown and Brown's [8] article gave the initial impetus to this paper. Our goal has been to formalise hypertext systems in order to compare their expressiveness. The Dexter model [9, 24] is generally regarded as a "standardising" document by the hypertext community. The Dexter model does not attempt to cover the details of user interaction with the hypertext. Our approach is different as we focus only on one possible type of interaction which is the follow link operation. Of course, other aspects of user interaction are useful, and could be modelled in future work.

Generally speaking, the Dexter model is far more detailed than our abstract machines. It can be regarded as requirements that have to be satisfied by implementations in ordered to be considered as hypertexts. However, unlike our approach, the Dexter model does not provide a formal criterion to compare hypertext systems.

In the Dexter model, links anchors have two parts: an *anchor id* and an *anchor value*; this design provides a fixed point of reference for use by the storage layer combined with a variable field for use by the within-component layer. This practical organisation provides for document editing, which is not an issue tackled in this paper.

Tompa [13] formalises hypertexts as graphs. He presents a data model, but does not define a runtime behaviour. He therefore cannot model functional links. However he gives a formal definition of user's views which, as far as links are concerned, could also be implemented with functional links. Doberkat [42] presents a Prolog-like specification language of hyperdocuments, whose execution yields a directed graph. Doberkat focuses on the compilation aspect, whereas we focus on the runtime interpretation of the graph.

Bieber and Kimbrough's bridge laws [2] are designed to generalise hypertexts; they provide a mechanism for automatic linking (as well as automatic creation of other components such as nodes or buttons). They regard link traversal as a complex inferencing process that may use application-level procedures. Such bridge laws are presented in a logic-based model.

Stotts, Furuta and Ruiz [43] present a browsing semantics of hypertext documents by examining links alone. They consider that links form an automaton instead of a static directed graph. Using branching time logics, they perform model checking on browsing traces in order to derive temporal properties of hypertext systems. Model checking proceeds by exhaustive exploration of the state space and therefore requires either a finite state space or a state space with a suitable finite representation. It is an open question to decide whether their model checking technique could also be applied to our automata, and in particular to functional links, because the latter may potentially generate infinite link instances.

Park [14] also studied dynamic properties of hypertext in terms of "readers' experience". He proved that the set of link followings in a hypertext, i.e. our hyperconnectivity, is regular. However, he does not investigate more powerful classes of automata.

Garg [44], and later Richard and Rizk [45] present models able to describe several hypertext systems. Garg [44] focuses on abstractions in hypertexts, while Richard and Rizk highlight the relationship between objects containing information. Furuta and Stotts [11] formalize hypertext systems in their Trellis model; their R-model [46] is more complete as it includes windows modelisation. Parunak describes hypertext systems in terms of their topologies [47].

Ashman [48] investigates different representations of links as relations. These representations are discussed in terms of four questions related to existence and identification of the source and destination anchors. A technique similar to the functional link is used as a possible link representation.

Mendes and Hall's goal [50] is complementary to our approach. They perform an empirical analysis of the power or usefulness of some hypertext features viewed from the authoring process. The advantage of our approach is that it allows us to isolate a given feature, here the linking paradigm, and to investigate its expressiveness independently of any other component. In the empirical approach, a complete hypertext system is used, and the difficulty is to determine the contribution of each component.

9. CONCLUSION

During our survey of linking mechanisms, we have defined three classes of hypertext systems. We have identified the class of regular hypertext systems: they are the least powerful and their power is equivalent to nondeterministic finite-state automata. History mechanisms give hypertext systems the power of pushdown automata. Functional links make hypertext systems as powerful as Turing machines.

This work can be continued in several directions. A number of hypertext notions can be incorporated in our abstract machines and in the set of observables; e.g. link attributes or types. Multi-windowing systems extend the notion of navigation, as they allow multiple documents to be in focus. The relationship between link following and general computation can also be investigated. Finally, multiple users could be included in the model, and issues of cooperative work could be addressed.

10. ACKNOWLEDGEMENT

We wish to thank Helen Ashman, Leslie Carr, Emilia Mendes, Peter Nürnberg, and the anonymous referees for their useful comments.

REFERENCES

[1] Judi Moline, Dan Benigni, and Jean Baronas, editors. *Proceedings of the Hypertext Standardization Workshop*, Nist Special Publication 500-178. National Institute of Standards and Technology, January 1990.

[2] Michael P. Bieber and Steven O. Kimbrough, (1994). On the Logic of Generalized Hypertext. *Decision Support Systems*, 11:241–257.

[3] Theodor Holm Nelson, (1987). *Literary Machines.* Project Xanadu.

[4] Vannevar Bush, (1945). As We May Think. *Atlantic Monthly*, (176):101–108.

[5] Douglas Engelbart, (1963). A Conceptual Framework for Augmenting Man's Intellect. *Vistas on Information Handling*, 1:1–29.

[6] Tim Berners-Lee and D. Connolly, (1995). Hypertext Markup Language Specification 2.0. Technical Report RFC 1866, MIT/LCS.

[7] Nicole Yankelovich, B. Haan, Norman Meyrowitz, and S. Drucker, (1988). Intermedia: The Concept and Construction of a Seamless Information Environment. *IEEE Computer*, pages 81–96.

[8] Peter J. Brown and Heather Brown, (1995). Embedded or Separate Hypertext Mark-up: Is It an Issue? *Electronic Publishing*, 8(1):1–13.

[9] Frank Halasz and Mayer Schwartz, (1994). The Dexter Hypertext. *Communications of the ACM*, 37(2):31–39.

[10] Brad Campbell and Joseph M. Goodman, (1988). HAM: A General Purpose Hypertext Abstract Machine. *Communications of the ACM*, 31(7):856–861.

[11] Richard Furuta and P. David Stotts, (1989). Programmable Browsing Semantics in Trellis. In *Proceedings of Hypertext'89*, pages 27–42.

[12] P. David Stotts and Richard Furuta, (1991). Dynamic Adaptation of Hypertext Structure. In *Proceedings of Hypertext'91.*

[13] Frank WM. Tompa, (1989). A Data Model for Fexible Hypertext Database Systems. *ACM Transations of Information Systems*, 7(1):85–100.

[14] Seongbin Park, (1998). Structural Properties of Hypertext. In *Proceedings of Hypertext'98*, pages 180–187, Pittsburgh, USA.

[15] Andrew M. Fountain, Wendy Hall, Ian Heath, and Hugh C. Davis, (1990). MICROCOSM: An Open Model for Hypermedia With Dynamic Linking. In A. Rizk, N. Streitz, and J. André, editors, *Hypertext: Concepts, Systems and Applications (Proceedings of ECHT'90)*, pages 298–311.

[16] Hugh Davis, Wendy Hall, Ian Heath, Gary Hill, and Rob Wilkins, (1992). Towards an Integrated Environment with Open Hypermedia System. In *Proceeding of the Second European Conference of Hypertext (ECHT'92)*, pages 181–190.

[17] Wendy Hall, Hugh Davis, and Gerard Hutchings, (1996). *Rethinking Hypermedia: the Microcosm Approach.* Kluwer Academic Publishers, Boston.

[18] Helen Ashman and Janet Verbyla, (1993). The Functional Model of the Link (poster presentation). In *Proceedings of Hypertext'93.*

[19] Helen Ashman and Janet Verbyla, (1994). Dynamic Link Management via the Functional Model of the Link. In *Proceedings of Basque International Workshop on Information Technology*, Biaritz, France.

[20] Matthias Felleisen, (1990). On the Expressive Power of Programming Languages. In *Proc. European Symposium on Programming*, number 432 in Lecture Notes in Computer Science, pages 134–151. Springer-Verlag.

[21] Gordon D. Plotkin, (1977). LCF Considered as a Programming Language. *Theoretical Computer Science*, 5:223–255.

[22] International Organization for Standardization, Geneva, Switzerland, (1986). *Information Processing, Text and Office Systems, Standard Generalized Markup Language (SGML) First edition.* International Standard ISO 8879-1986. Federal information processing standard; FIPS PUB 152.

[23] Hanan Samet, (1984). The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surveys*, 16(2).

[24] Kaj Grønbæk and Randall H. Trigg, (1996). Toward a Dexter-based Model for Open Hypermedia: Unifying Embedded References and Link Objects. In *The Seventh ACM Conference on Hypertext (HT'96)*, pages 149–160, Washington DC.

[25] Harry R. Lewis and Christos H. Papadimitriou, (1981). *Elements of the Theory of Computation.* Prentice-Hall.

[26] Steven R. Newcomb, Neill A. Kipp, and Victoria T. Newcomb, (1991). HyTime: the Hypermedia/Time-based Document Structuring Language. *Communications of the ACM*, 34(11):67–83.

[27] Tim Bray and S. De Rose, (1997). Extensible Markup Language (XML): Part 2. linking. Technical report, WWW Consortium. Available from `http://www.w3.org/TR/WD-xml-link`.

[28] Danny B. Lange, (1990). A Formal Model of Hypertext. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 145–166. National Institute of Standards and Technology.

[29] Kasper Øterbye and Uffe Kock Wiil, (1996). The Flag Taxonomy of Open Hypermedia Systems. In *Proceeding of the Seventh ACM Conference on Hypertext (HT'96)*, Washington.

[30] Tim Berners-Lee, Robert Cailliau, A. Luotonen, Henrik Frystyk Nielsen, and A. Secret, (1994). The World Wide Web. *Communications of the ACM*, 37(8):76–82.

[31] Hugh Davis, (1995). *Data Integrity Problems in an Open Hypermedia Link Service.* PhD thesis, University of Southampton.

[32] Paul Thistlewaite, (1997). Automatic construction and management of large open webs. *Information Processing and Management*, 33(2):161–173.

[33] Michael Bieber and Jiangling Wan, (1994). Backtracking in a Multiple-Window Hypertext Environment. In *ACM European Conference on Hypermedia Technology ECHT'94*, pages 158–166, Edinburgh, UK.

[34] Aggelos Pikrakis, Tilemahos Bitsikas, Stelios Sfakianakis, Mike Hatzopoulos, Dave DeRoure, Wendy Hall, Sigi Reich, Gary Hill, and Mark Stairmand, (1998). Memoir — software agents for finding similar users by trails. In *The Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents (PAAM'98)*, London, UK.

[35] Renaud Deschamps, (1995). *Bases de connaissance généralisées: une approche fondée sur un modèle hypertexte expert.* PhD thesis, Université Paul Sabatier de Toulouse.

[36] R. Pausch and J. Detmer, (1990). Node Popularity as a Hypertext Browsing Aid. *Electronic Publishing*, 3(4):227–234.

[37] Janet Verbyla and Helen Ashman, (1994). A User-Configurable Hypermedia-Based Interface via The Functional Model of the Link. *Hypermedia*, 6(3):193–208.

[38] Thomas Kirste, (1996). A Functional Object Model for Hypertext Providing Definitional Transparency for Navigation Functions. Technical report, Computer Graphics Center.

[39] Jocelyne Nanard and Marc Nanard, (1995). Hypertext Design Environments and the Hypertext Design Process. *Communications of the ACM*, 38(8):49–56.

[40] Manfred Thuring, Jorg Hannemann, and Jorg M. Haake, (1995). Hypermedia and Cognition: Designing for Comprehension. *Communications of the ACM*, 38(8):57–66.

[41] Luc Moreau, (1996). Correctness of a Distributed-Memory Model for Scheme. In *Second International Europar Conference (EURO-PAR'96)*, number 1123 in Lecture Notes in Computer Science, pages 615–624, Lyon, France. Springer-Verlag.

[42] Ernst-Erich Doberkat, (1996). A Language for Specifying Hyperdocuments. *Software–Concpets and Tools*, 17:163–172.

[43] P. David Stotts, Richard Furuta, and J. Cyrano Ruiz, (1992). Hyperdocuments as Automata: Trace-Based Browsing Property Verification. In *Proceeding of te ACM Conference on Hypertext*, pages 272–281.

[44] Pankaj K. Garg, (1988). Abstraction Mechanisms in Hypertext. *Communications of the ACM*, 31(7):862–870.

[45] Gilles Richard and Antoine Rizk, (1990). Quelques Idées pour une Modélisation des Systèmes Hypertextes. *Technique et Science Informatiques*, 9(6):505–514.

[46] Richard Furuta and P. David Stotts, (1990). A Functional Meta-structure for Hypertext Models and Systems. *Electronic Publishing*, 3(4):179–205.

[47] H. Van Dyke Parunak, (1989). Hypermedia Topologies and User Navigation. In *Proceedings of Hypertext '89*, pages 43–50.

[48] Helen Ashman, (1998). Relations modelling sets of hypermedia links and navigation. *Submitted to the Computer Journal.*

[49] Michael P. Bieber and Tomas Isakowitz, (1994). Text Editing and Beyond, a Study in Logic Modeling. *Decision Support Systems*, 11:219–240.

[50] M. Emilia X. Mendes and Wendy Hall, (1997). The SHAPE of Hypermedia Authoring for Education. In *Proceedings of ED-MEDIA & ED-TELECOM 97*, pages 877–885, Calgary, Canada.