FOHM: A Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains

Dave E. Millard, Luc Moreau, Hugh C. Davis
Multimedia Research Group
Department of Electronics & Computer Science
University of Southampton
SO17 1BJ, UK
phone +44 1703 59 3255
{dem97r, lavm, hcd}@ecs.soton.ac.uk

Siegfried Reich
Department of Information Systems
University of Linz, 4040 Linz, Austria
sre@ifs.uni-linz.ac.at

ABSTRACT
The Open Hypermedia Systems community has been largely concerned with interoperability between hypertext systems which share the same paradigm. It has evolved a component based framework for this purpose, in which specific but incompatible middleware components are designed for each hypertext domain, such as navigational hypertext, spatial hypertext or taxonomic hypertext. This paper investigates the common features of these domains and introduces FOHM, a **F**undamental **O**pen **H**ypertext **M**odel, which defines a common data model and set of related operations that are applicable for all three domains. Using this layer the paper explores the possible semantics of linking between different hypertext domains, and shows that each can introduce features which benefit the other domains.

KEYWORDS: Component-based Open Hypermedia System (CB-OHS), Open Hypermedia Protocol (OHP), Hypertext Domains, Fundamental Open Hypermedia Model (FOHM), Interoperability.

INTRODUCTION
At the First Open Hypertext Systems (OHS) Workshop held at Hypertext '94, Antoine Rizk proposed that the workshop should design a protocol for communication between hypertext client programs and link servers [30]. The motivation for his suggestion was that the OHS community was wasting much time re-implementing client programs and wrappers for existing third party applications. The premise of his proposal was that there were many features that were common to all hypertext systems, and that if we could produce a simple protocol that all clients used, then we would have interoperability of clients, and we could concentrate on research into link services and hyperbases.

This goal seemed reasonable, and in 1996, Davis et al. reported back to the second OHS workshop with a draft proposal for such a protocol [4][17]. This proposal leant heavily on the authors' own experiences in designing Microcosm and Multicard, and it had a few unfilled sections that the authors hoped that the Workshop would help to fill. However, when viewed by a wider community the draft proposal was seen to have many limitations [1]. One of the main problems was that the original protocol dealt only with a quite specific model of point to point navigational hypertext, and had no way of dealing with co-operation, distribution, or other models of hypertext such as taxonomic hypertext [19], spatial hypertext [10] or features such as transclusions in Xanadu [14].

As a result the community moved on from a straightforward client/server approach to a component based approach [29]. In this architecture, component-based middleware is implemented for each hypertext domain and is responsible for mapping the specific structures required by the client into and out of the structures stored in an all purpose back-end server. This model provides an extensible architecture where new hypertext domains can be added by defining a new interface for this domain, and implementing an appropriate client and middleware component. This is fully reported in Reich et al. [21] and has been the subject of demonstrations in recent hypertext conferences, where components and systems from different research labs have been seen to interoperate.

Having designed an architecture which allows multiple hypertext domains to operate alongside each other within a single framework, the community has recently become interested in the possibility of interoperability between the different hypertext domains themselves.

There are two distinct sorts of inter-domain interoperability that one might imagine. Let us imagine that there are two workspaces, the first which is a traditional navigational hypertext workspace, and the second which is a spatial hypertext. If there were a link from a node in the first workspace to a node in the second workspace, what would be the semantics of following this link? The first and perhaps most obvious outcome would be that as the link was followed into the spatial workspace, a spatial browser would open from where the user could continue to browse. However an interesting second possibility could be that the navigational browser could remain open, and could continue to attempt to interpret the spatial workspace *as if it was* a navigational workspace. It was this second interpretation of inter-domain interoperability [16], that engendered to the research reported in this paper.

In order for one hypertext domain to be able to interpret a second hypertext domain as if it were the first, then, using the current architecture, each middleware component must be adapted so that it can translate structures in all other domains into its own domain, and this solution will exhibit quadratic growth with the number of domains that intend to interoperate. Alternatively we could attempt to define a *common* storage layer which would be sufficiently generic that it could store the structures required by all current domains, and hopefully all future domains.

In this paper we start by describing the characteristics, similarities and differences between three of the existing hypertext domains (navigational, spatial and taxonomic). We then describe FOHM, a fundamental open hypertext model, which identifies data and permitted operations in a client/server style architecture. A formal specification of these operations and data is presented. We demonstrate that this model is sufficient to represent *all* the structural abstractions and operations of these three domains. We make the conjecture that our model is general enough to support other types of domains. Although it is not *formally* possible in the generic case to define a mapping back from the storage layer to any other domain, we discuss ways in which we can attempt this mapping, and we conclude by discussing the consequences of achieving such mappings, and the advantages and disadvantages that domains can introduce to each other.

HYPERTEXT DOMAINS

The application of hypertext is often seen in navigating information spaces with the World Wide Web serving as a prominent example. However, there are many more hypertext domains, each with their own specific needs and requirements. These domains include spatial hypertext, argumentation support, taxonomic hypertext, hypermedia art, hypermedia literature and others (for an overview see Nürnberg, 1997 [15]).

These hypertext domains and the systems supporting them rely on different conceptual models [27]. The ultimate overall objective common to all these conceptual models is to “understand” information spaces [24]. Different criteria for measuring this objective may apply, e.g. metrics borrowed from software engineering such as “cohesion” and “coherence” [9], and the cognitive requirements of these domains differ. However the domains could still be used together. For example, users navigating the Web may on arrival at a “new server” start by having a spatial overview of the set of pages (nodes) available (e.g. by using a graphical map) before navigating them. Equally, after having read a set of nodes in a navigation browser they may want to use a taxonomic tool in order to reason about the set of nodes they have read. For example to determine whose opinions academic papers represent.

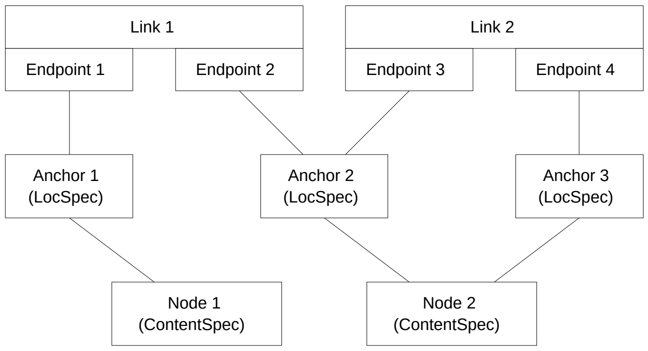

Figure 1: The OHP Node Link model

In the following we will briefly describe some of these domains and reflect on their specific properties. The domains described are navigational hypertext, spatial hypertext and taxonomic hypertext.

The Navigational Domain

Navigational hypertext is concerned with partitioning information spaces into nodes and establishing relationships between them such that users can navigate them. The concept is simple but powerful and usable. Links store connections between “hot spots” in documents. By clicking on one hotspot the user navigates to the one at the other end of the link. It is probably the oldest conceptual model of hypertext and is already mentioned by the pioneers [2, 6, 13]. Modern Open Hypermedia Systems (OHSs) combine many features of early systems, keeping linking information separate from documents and allowing for more powerful link structures, for example bi-directional or n-ary links.

The model that we present here follows closely the data model specified by the Open Hypermedia Systems Working Group (OHSWG) [21]. It is based firmly on the definition of several important objects that make up a link structure. Consider the two links shown in Figure 1.

In this diagram there are two node objects. These represent the systems notion of a document or a file. Each node may have several anchors associated with it. These are objects that define a region inside the node to use as a hotspot (for example there could be an anchor from word twelve to word seventeen). The endpoint objects bind an anchor to a particular link. As an anchor can be bound to several links (see anchor 2 in the Figure) the endpoint contains all the information that is relevant to this anchor in the context of one particular link.

Operations applied to objects include creation and deletion primitives for all the objects in the system as well as a Follow Link function that returns a set of endpoints based on a single input endpoint parameter according to the underlying link structures.

The Spatial Domain

This domain of hypertext is also referred to as "Information Analysis' [9] or "Information Triage" [11]. It emerged from activities like collecting, comprehending or interpreting diverse sets of information and the need to support such activities. Thus, as opposed to navigational hypertext, what matters is the ability to create and move nodes freely. The key characteristic is to leave structure implicit and informal (at least as presented to the user) [10].

Relationships between nodes are simply expressed by their visual characteristics such as spatial proximity, color or shape. This results in some interesting properties. If for instance a node is slightly misaligned with other nodes then this might express an uncertainty about whether this node is actually part of this relationship. In other words it expresses classification within relationships, where some nodes are 'more' related then others. Spatial hypertext systems are therefore inherently flexible. Examples of spatial hypertext systems include VIKI [10] and CAOS [22].

The following conceptual model therefore summarizes spatial hypertext. Nodes are visual symbols servicing as wrappers to documents, therefore they have characteristics such as color, location and shape. Nodes can be aggregated (visually) thus building collections of related objects. Composites are the visual representations of these collections, they also have visual characteristics and therefore can also be aggregated into other composites, although this relationship may not be circular. These collections are typed, i.e. they may form lists, matrixes, sets, stacks etc. This effects their visual presentation but also acts as an organizational aid, adding both order and internal structure to the collections (i.e. one node may be placed before another, or the proximity of one node to another could reflect the strength of their association).

Any spatial system has to make all of these relationships explicit in the system so that queries can be made of the information and that visual information can be stored economically. To this end many systems use spatial parsers to convert the implicit spatial relationships into explicit associations within the system. It is this parser that recognizes the way in which nodes have been lain down and decides on an appropriate structure to store the information such as a list or a set.

Operations on objects include creation and deletion, adding and removing objects to composites and also a 'zoom' function that allows a user to zoom into a composite's subcomponents.

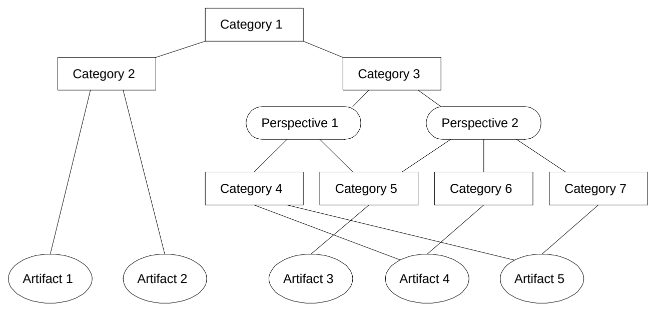

Figure 2: Example Taxonomy

The Taxonomic Domain

Taxonomic hypertext applies hypertext concepts such as nonlinear information access and individualized views to the domain of reasoning about taxonomies, e.g. in biology, linguistics and other application areas [19, 27, 28].

For this kind of knowledge task a model based on set theory is used [27]. Users sort artifacts (the equivalent of nodes) into categories (sets) based on their characteristics. Different users may have different views of how the artifacts are partitioned. Taxonomic reasoning is the process of moving around the structure by crossing the boundaries between overlapping sets.

Figure 2 gives an example of a Taxonomy. Here two people have categorized five artifacts. They both agree that all five lie within category one and that artifacts one and two also lie within sub-category two. However they disagree on how the remaining three artifacts should be split up. The first view (perspective one) says that artifacts four and five lie in category four while artifact three lies in category five. The second view (perspective two) agrees that artifact three should be in category five but thinks that artifacts four and five are different enough to be categorized separately (in categories six and seven respectively).

There are two important rules that govern the shape of taxonomies.

1. Whenever a taxonomy splits via perspectives the same artifacts can be reached down each branch, it is merely their categorization that changes.

2. All categories are single parented within a single taxonomy (i.e., an artifact may be in two categories only when each parent category is within a different taxonomy).

Artifacts can be added to categories and also removed. Categories are organized hierarchically so categories may be added and removed as well. Categories cannot be circular, as this is semantically meaningless [27]. In addition some set like operations are also required so as to reason about multiple taxonomies, such as set union and intersection.

Comparison of Domains

There are a number of similarities between the different hypertext domains: in all domains, we find notions of *data* (node in navigational, visual in spatial, or artifact in taxonomic hypertext) or *association* (link in navigational, composite in spatial or category in taxonomic hypertext). On the other hand, there are features that only appear in a single specific hypertext domain:

- The navigational domain contains anchors that allow links to point into documents rather then be anchored on entire documents.
- The spatial domain introduces typed structures (e.g. lists, sets, etc.) to the relationships.
- The taxonomic domain contains perspective objects that allow the relationship structures to diverge according to different views.

Other differences concern restricting the possible composite structures for various domains. For example circularity is allowed in navigational hypertext but not in taxonomic or spatial. A more elaborate comparison of hypertext domains based on different features can be found in [15].

In the following section we will introduce a formal model of these domains which we shall then use to explore the common ground between them.

FORMALIZATION

What is a Formalization of a Hypertext Domain?

There are numerous formal models of hypertexts, but they differ according to their motivation. The Dexter model [8] defines a common vocabulary and its meaning in order to talk about hypertext systems. The hypertext abstract machine (HAM) [3] is an architectural description of a general-purpose, transaction-based, multi-user server for a hypertext storage system. Furuta and Stotts' Trellis model [7] is a formal specification of hypertexts based on petri nets. Hypertexts have also been formalized as graphs [25] or automata [12, 20].

Several authors [12, 20, 23] have studied the dynamic properties of hypertexts in terms of "reader's experience", formalizing what readers see when they interact with a hypertext system.

In this paper, our motivation is to make data structures and operations pertaining to them explicit, so that we can define a model that is common to the different hypertext domains and investigate interoperability.

Since both data and operations have to be formalized, we adopt the same approach as Moreau and Hall [12] for our formalization of a hypertext domain. We define an *abstract machine* characterizing the states of hypertext clients and servers. We define the client as the browser, or user interface, displaying documents, whereas the server maintains the documents and all hyperstructures in its hyperbase. Supported operations are modeled by transitions of the abstract machine, possibly changing the internal states of the client and the server. In a first instance, we do not make distribution explicit, and we consider only one server in the system.

Abstract Machine Transition

Let us consider the navigational domain. The first step in the formalization is to define the *state space* of the abstract machine. The state space not only comprises the *data model* but also the *architectural organization*. The data model defines the different data that can be encountered: node, anchor, endpoint and link. The architectural organization is composed of a *store* containing all hyperstructures and the state of the client, which is the currently displayed document. Multiple windows, history lists, etc. are also possible to model but are beyond the scope of this work.

The second step in the formalization is to define the *operations* that may be performed: they include creating links (or any other data), retrieving data from the store, or even performing a query such as getting endpoints for a given data.

Given a state S_1^N of the navigational abstract machine, executing an operation leads to a new state S_2^N, which we note as:

$$S_1^N \mapsto_N S_2^N,$$

where $\mapsto_N$ denotes a transition following an operation in the navigational domain. Each hypertext domain may be formalized in a similar manner, which gives us transition relations $\mapsto_S$ for spatial hypertext and $\mapsto_T$ for taxonomic hypertext.

A Fundamental Hypertext Layer

Given the similarities between the different domains we have defined a *fundamental layer* as composed of data structures that are general enough to "encode" any data structure of each hypertext domain. These operations and data form a fundamental open hypertext model, which we call FOHM. We also define general operations on these data structures. Figure 3 contains the data model, whereas Figure 4 displays some operations on these data, which we now describe.

The set of *identifiers* $\mathcal{I}$ is formed by the union of three sets of identifiers, respectively for associations $\mathcal{I}^a$, data $\mathcal{I}^d$, and data references $\mathcal{I}^r$ (abbreviated refs). *Data*, typically documents, are not formalized in the hypertext layer; however, as we need to refer to them, we assume the existence of a set $\mathcal{D}$ of Data. The set $\mathcal{A}$ of *associations* is defined by the cartesian product of three sets, respectively of binding vectors $\vec{\mathcal{B}}$, relation types $\mathcal{T}$ and structural types $\mathcal{S}$. A *binding* is composed of a ref ID and a vector of features; the former is meant to be the identifier of a reference, which "attaches" on a document. A *vector of features* can be regarded as an ordered set of properties, where each property identifies a value in a set of features, such as direction of a binding, geometric property, etc. By construction, we require the vector of features of all the bindings in a given association to have the same dimension as the feature space vector in the association. The

$$\begin{array}{rcll}
\mathcal{I}^a & = & \{\alpha_0^a, \alpha_1^a, \ldots\} & \text{(Association IDs)} \\
\mathcal{I}^d & = & \{\alpha_0^d, \alpha_1^d, \ldots\} & \text{(Data IDs)} \\
\mathcal{I}^r & = & \{\alpha_0^r, \alpha_1^r, \ldots\} & \text{(Refs IDs)} \\
\mathcal{I} & = & \mathcal{I}^a + \mathcal{I}^d + \mathcal{I}^r & \text{(IDs)} \\
\mathcal{D} & = & \{d_0, d_1, \ldots\} & \text{(Data)} \\
\mathcal{B} & = & \mathcal{I}^r \times \vec{\mathcal{F}} & \text{(Bindings)} \\
\vec{\mathcal{B}} & = & \mathbb{N} \to \mathcal{B} & \text{(Binding Vectors)} \\
\mathcal{A} & = & \vec{\mathcal{B}} \times \mathcal{T} \times \mathcal{S} & \text{(Associations)} \\
\mathcal{T} & = & Name \times \vec{\mathcal{N}} & \text{(Relation Types)} \\
\mathcal{S} & = & \{\mathsf{heap}, \mathsf{list}, \mathsf{stack}, \mathsf{perspective}, \ldots\} & \text{(Structural Types)} \\
\mathcal{R} & = & \mathsf{whole} : \mathcal{I} \to \mathcal{R} \mid \mathsf{inside} : \mathcal{I} \times \mathcal{L} \to \mathcal{R} & \text{(Data Refs)} \\
\mathcal{L} & = & \{l_0, l_1, \ldots\} & \text{(Loc Specs)} \\
\mathcal{F} & = & Dir\ +\ Shape\ +\ \ldots & \text{(Features)} \\
\vec{\mathcal{F}} & = & \mathbb{N} \to \mathcal{F} & \text{(Feature Vectors)} \\
Dir & = & \{\mathsf{src}, \mathsf{dst}, \mathsf{bi}\} & \text{(Direction Features)} \\
Shape & = & \{\mathsf{square}, \mathsf{circle}, \ldots\} & \text{(Shape Features)} \\
\mathcal{N} & = & \{\mathsf{direction}, \mathsf{shape}, \ldots\} & \text{(Features Spaces)} \\
\vec{\mathcal{N}} & = & \mathbb{N} \to \mathcal{N} & \text{(Feature Space Vectors)} \\
Name & = & String & \\
Object & = & \mathcal{D} + \mathcal{A} + \mathcal{R} & \text{(Objects)} \\
Store & = & \mathcal{I} \to Object & \text{(Store)}
\end{array}$$

Figure 3: The Fundamental Data Model

feature space vector enumerates, by name, the different properties that must be defined in each binding of an association: this allows clients to construct bindings that are understood by servers and other clients. We can regard the vector of features as an *extensible record*, whose components are identified by the feature space vector. The extensible record is a critical technique to support other hypertext domains, as it allows us to implement data structures with a variable number of features. Finally, an association is also characterized by a structural type, such as heap, list, or stack, which exhibit different behaviors when the association is traversed.

We define the set *Object* as the union of Data, Associations and Refs. The *store*, which is defined as the permanent memory of a server, maps identifiers onto objects.

Figure 4 displays some of the operations that are permitted on the data: for instance, *createAssociation* expects an association and a store, and if successful, returns a new identifier for the association, and an updated store, containing a mapping between the identifier and the association. As a result, *retrieving* an object associated with a valid identifier is performed by applying the store to the identifier. Along the same lines, adding a binding to an association (*addBindingToAssociation*) consists of retrieving the association and storing a new association with a new binding in the binding vector.

Note that the specification of all functions in Figure 4 is *executable*. There are however operations whose specification is not executable, which we call *queries*. The specification constrains the type of results such queries should produce, but it does not indicate how the result should be obtained because we do not want to preclude any implementation. For instance, the query *getBindingForData* returns bindings with a given feature vector $\vec{f}$ and whose Data Ref refers to a given identifier i.

It is possible to formalize a complete abstract machine, composed of a client executing a sequence of operations and a server containing a store. Transitions will be defined in terms of the supported operations, and their effect on the client and server is defined using the definitions of Figure 4. As a result, we can define a transition relation for the fundamental hypertext machine, which we note $\mapsto_F$.

A Translation to the Fundamental Hypertext Layer

The claim that the fundamental layer is general enough to encode any of our three selected hypertext domains may itself be formalized. For each hypertext domain, there exists a *translation* that converts any state of the abstract machine into a configuration of the fundamental hypertext abstract machine. Similarly, for each hypertext domain, each transition of the abstract machine may be translated into one (or more) transition(s) of the fundamental hypertext abstract machine.

For the navigational domain, the translation function is noted T_N, and given a navigational state S^N, the expression $T_N[\![S^N]\!]$ denotes the corresponding fundamental state.

Examples: $i)$ A *link* of the navigational domain is defined as an association with a single feature space: a direction. Each endpoint is mapped onto a binding whose feature vector contains a single entry whose value is in the set Dir. $ii)$ A

Functions:

$$
\begin{array}{rcl}
createAssociation & : & \mathcal{A} \times Store \to (\mathcal{I}^a \times Store)_\perp \\
 & = & \lambda\, a\, \sigma.\ \textbf{let}\ i\ =\ newId(\sigma) \\
 & & \qquad\quad \textbf{in}\ (i, \sigma[i \to a]) \\
 & & \qquad\quad \textbf{end} \\
deleteAssociation & : & \mathcal{I}^a \times Store \to Store_\perp \\
 & = & \lambda\, i\, \sigma.\sigma[i \to \perp] \\
 & & \\
retrieve & : & \mathcal{I} \times Store \to Object_\perp \\
 & = & \lambda\, i\, \sigma.\sigma(i) \\
 & & \\
addBindingToAssociation & : & \mathcal{I}^a \times \mathcal{I}^r \times \vec{\mathcal{F}} \times Store \to Store_\perp \\
 & = & \lambda\, i^a\, i^r\, \vec{f}\, \sigma.\ \textbf{let}\ \ (\vec{b}, t, s) \ = \ retrieve(i^a, \sigma) \\
 & & \qquad\qquad\qquad\qquad\ \ \vec{b'} \ = \ \vec{b}[size(DOM(\vec{b})) \to (i^r, \vec{f})] \\
 & & \qquad\qquad\quad \textbf{in}\ \sigma[i^a \to (\vec{b'}, t, s)] \\
 & & \qquad\qquad\quad \textbf{end}
\end{array}
$$

Queries:

$$
\begin{array}{rcll}
getBindingForData & : & \mathcal{I} \times \vec{\mathcal{F}} \times Store \to (SetOf(\mathcal{B} \times \mathcal{I}^a))_\perp & \\
(i, \vec{f}, \sigma) & \to & \{(b_1, i_1^a)\ :\ (\mathcal{B} \times \mathcal{I}^a)\ | & b_1 = (i_1^r, \vec{f_1}) \\
 & & & b_1 \in retrieve(i_1^a, \sigma).\vec{B} \\
 & & & \vec{f_1} = \vec{f} \\
 & & & retrieve(i_1^r, \sigma).I = i\ \ \}
\end{array}
$$

Figure 4: Some Fundamental Functions and Queries

composite of the spatial domain is defined by an association with a three dimensional vector space for shape, color and location. There is no direct equivalent of a binding in the spatial domain, however, the translation creates such a binding, with a feature vector containing the shape, color and location of the data in the composite.

There is a strong relationship between the transition relations that we defined for each hypertext domain and the translation to the fundamental layer. It takes the form of a *diamond property* property, which we can state as follows. Let S_1^N, S_2^N be two states of the navigational domain and S_1^F a state of the fundamental one. If there is a translation from S_1^N to S_1^F, and if there is a transition from S_1^N to S_2^N, then there exists S_2^F, such that there is one (or more) transitions from S_1^F to S_2^F, and S_2^F is the result of the translation of S_2^N into the fundamental layer. Such a property is summarized by the following commutative diagram:

$$\begin{tikzcd}
S_1^N \arrow[r, phantom, "\mapsto_N"] \arrow[d, "T_N"'] & S_2^N \arrow[d, "T_N"] \\
S_1^F \arrow[r, phantom, "\mapsto_F^+"] & S_2^F
\end{tikzcd}$$

Similar properties also hold for the spatial and taxonomic hypertexts.

Intuitively, this property states that the fundamental layer is expressive enough to encode any of the three hypertext domains and *simulate* its behavior. We make the conjecture that our model is general enough to simulate other hypertext domains: in particular, extensible records can accommodate features that are not present in the three investigated domains.

A Fundamental Implementation Technique

Interestingly, each translation to the fundamental layer is reversible. Let us call T_N^{-1} the inverse translation of T_N. The following property holds: for any state S^N of the navigational hypertext machine,

$$T_N^{-1}[\![T_N[\![S^N]\!]]\!] = S^N.$$

A similar property holds for the other translations from spatial and taxonomic. Consequently, the above commutative diagram may be rewritten as follows.

$$\begin{tikzcd}
S_1^N \arrow[r, phantom, "\mapsto_N"] \arrow[d, "T_N"'] & S_2^N \\
S_1^F \arrow[r, phantom, "\mapsto_F^+"] & S_2^F \arrow[u, "T_N^{-1}"']
\end{tikzcd}$$

This diagram proves that the fundamental layer can be used to *implement* any hypertext domain. Given a state, it suffices to convert it to the fundamental representation, to perform the operation in the fundamental layer, and to convert the state back to the initial domain.

Note that we do not recommend that any hypertext is implemented by this complex double translation; however, this result shows that:

1. behaviors of any domain can be explained in terms of the fundamental layer,
2. execution in the fundamental layer suffices, and domain specific information has to be converted only when made visible to the user:

$$\begin{tikzcd}
S_1^N \arrow[r, mapsto, "N"'] & S_2^N \\
S_1^F \arrow[u, "T_N^{-1}"] \arrow[r, mapsto, "+"', "F"] & S_2^F \arrow[u, "T_N^{-1}"']
\end{tikzcd}$$

Investigating Interoperability

Note that the symmetric property $T_N[\![T_N^{-1}[\![S^F]\!]]\!] = S^F$ does not hold because the inverse translation T_N^{-1} is not defined for any fundamental state S^F. However, the inverse translation is the route to interoperability between hypertext domains.

Inverse translations may be extended to support data that come from different domains. It then becomes possible to define the meaning of domain-specific operations on data that do not belong to that domain. For instance, mapping a navigational hypertext to spatial hypertext is defined as follows:

$$T_S^{-1}[\![T_N[\![S^N]\!]]\!]$$

but requires extending T_S^{-1} to objects of the navigational domain.

INTEROPERABILITY BETWEEN DOMAINS

We have already stated that there are several features that only appear in any one of the three domains. The key to creating an inverse translation is to define corresponding features in the other two domains. These features should not be arbitrary but meaningful; bringing something to the domain that was not previously present. In this section, we suggest cross-domain translations of data-structures and their consequences.

Navigational Anchors

Anchors allow navigational clients to reference parts of documents that they would otherwise have to refer to in their entirety. This is particularly useful when defining hotspots but can also be useful when defining destination points, particularly in large documents or temporal media. The notion of an anchor could easily be extended to the other domains. Allowing parts of larger documents to be organized spatially or taxonomically and improving the granularity of referencing.

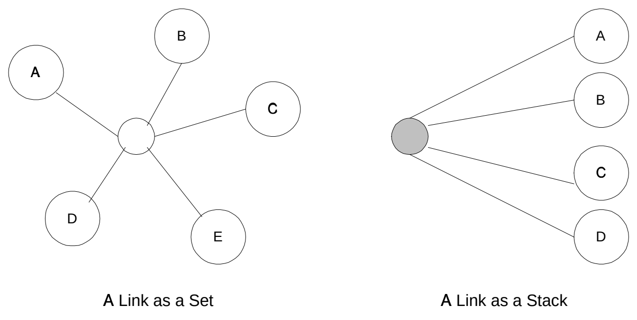

Figure 5: Link Structures

Spatial Structures

Spatial structures are different from other associations in that they contain internal structure. This provides the ability to produce sequences of nodes (using queues, lists and stacks) as well as maps that provide information about the distance of one link endpoint from another (through the use of matrixes). In taxonomic hypertext this would become another distinguishing feature of a category, allowing perspectives to split over structure as well as content. In navigation it could provide more structured movement through the information space. Navigational hypertext involves associations in two ways: traversal and arrival and each would be effected by internal structure.

Traversal is the act of following a link [26]. At the moment a link is effectively a set, if it could have different internal structure then the behavior of the traversal could vary according to that structure. Arrival is the act of viewing a structure (for example at the end of a traversal). Here internal structure could effect the way in which members of the association are viewed. Table 1 shows some possible semantics for traversal and arrival over different structures.

These structural types would add valuable organization to otherwise disorganized hyperwebs. For example take a look at Figure 5.

In Figure 5, the left hand link is represented as a Set, this means it is unordered but that no endpoint (A, B, C, D and E respectively) appears more then once. The right hand link is represented as a Stack; similar to a Set except it contains some internal structure, such that endpoint A is on the top of the stack while endpoint D is on the bottom. Our link semantics dictate that when users arrive at a Set they see all of the endpoints contained, thus if users arrive at the left hand link they see all the endpoints A through E. However, when they arrive at a Stack they see only the top endpoint of the stack, so in the case of the right hand link, this means they see endpoint A.

When traversing a link the user always uses an endpoint that is a member of that link as a starting point. When traversing a Set the user arrives at all the endpoints except the starting one. So if the user started from endpoint B on the left hand

Structure	**Traversal: endpoints user reaches**	**Arrival: object user sees**
Set	All except starting endpoint	All members
Stack	Endpoints to either side	Top of stack
Queue	Next endpoint in queue	Start of queue
List	Endpoints to either side	All members
Matrix	Nearest endpoints	All members

Table 1: Semantics of Traversal and Arrival

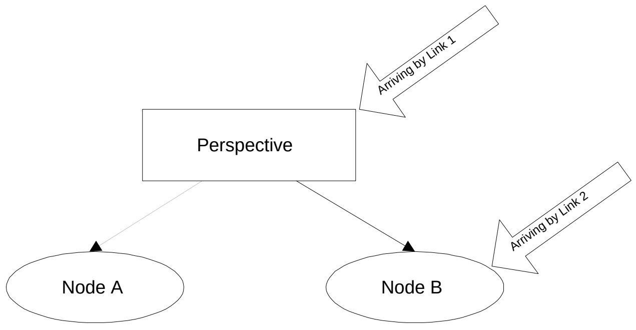

Figure 6: A Perspective over two Nodes

link then they would arrive at endpoints A, C, D and E. However, when a user traverses a stack they arrive at the endpoints to either side of their starting point. So if they started from endpoint B on the right hand link then they would arrive only at endpoints A and C.

Taxonomic Perspectives

Taxonomic perspectives allow different views of the organizations of some set of nodes within a branch of a hierarchy. Perspective is an abstraction that is in many ways similar to that of “context” as used in some navigational hypertext systems. Although a definition of context is beyond the scope of this paper (for an overview see [5]), it is generally understood that data (nodes and links etc.) will be arranged in a number of different contexts, and that the arrangement that will be seen by a user will depend in some way on a model that is known about this user, for example, what they have seen before, or their skill level.

Modeling perspectives as associations has the interesting property of allowing users to move between the different views just as if you were following a link. It also provides a convenient way of choosing a view. For example, in Figure 6, arriving by link 2 takes you directly into a particular view, whereas arriving by link 1 takes you to the perspective itself, where the system (or the user) can decide which view to provide. These are very much the properties that are required by context in other hypertext systems.

FOHM does not currently have any model of context, but these observations indicate the need for this extension, and demonstrate the way in which the features of one domain can add to and improve the features in another.

SUMMARY AND CONCLUSIONs

Inter-domain interoperability is a relatively new concern to the hypermedia community. This paper has presented one solution involving the definition of FOHM, a common model capable of representing structures and implementing operations from any of the three domains presented. Thanks to our use of extensible records, we make the conjecture that FOHM is general enough to support other hypertext domains.

This is somewhat in contrast to [21] where a dynamic arrangement of middleware components (known as structure servers) are responsible for dealing with the differing semantics of various domains stored in a storage layer. The advantage of this is that the system is able to cope with a wide range of different domains and adapt to the introduction of new ones. However, because it is not known which structure servers will be built on top of the storage layer the interface between the two has to be very general and system performance can suffer as a result. FOHM avoids these problems because it restricts itself to three common domains and is therefore able to ”know” about certain fundamental properties (such as structure type). An additional advantage is that frontend entities serving one domain (e.g. navigational hypertext) are to a certain extent able to reason about other domains (e.g. spatial hypertext).

In particular FOHM captures the hypertext structure of the various domains so that middleware components become optional rather then a necessity and clients inherit the responsibility of fulfilling structural requirements. For example a link created in a navigational client would need to be interpreted as a set by a spatial browser. In FOHM no interpretation is needed as the concept of association structure (lists, sets etc.) is captured in the model. In other words both navigational and spatial browsers know about sets and therefore the navigational client would already have added this itself.

Non-critical information that has not been captured in the model could still be dealt with by middleware. Presentation information, such as color, is a particularly good example. In this case it is the responsibility of either the client or the server to use sensible defaults to fill in information that it requires but that is not stipulated in the model. Such information is dependent on that particular client or server (for example a client that displays only greyscale) and has no place in the model presented here.

We have also shown that the exercise of defining a common model for these selected hypertext domains potentially results in a type of "cross fertilization" in that domains may benefit from each other. For instance, structure types as required in spatial hypertext may well be applied to navigation or taxonomic hypertext as well. Equally, perspectives as known from taxonomic hypertext contribute to both the other domains. It is particularly interesting to note that the functionality added seems to reflect many of the features previously worked into hypermedia systems. For example perspectives seem to reflect the need for context and when viewed in a navigational browser lists appear to be very similar to the notion of a tour or a trail.

We believe that the fact that FOHM reflects these requirements is very encouraging and intend to extend the formal definition to include the inverse mappings described. In addition a comprehensive set of structural types needs to be decided and the semantics of link traversal and arrival on these structures defined.

As a practical proof of concept a navigational and a spatial browser have been implemented, both of which use the FOHM layer to access a common set of structures within a FOHM server. It is envisaged that a taxonomic browser will also be written with the intention of exploring the quality and validity of macro-structures (such as a whole taxonomic hierarchy) that have been through a map-inverse map process.

The potential of FOHM to be included in the component based approach also needs to be evaluated. For example it could build a basis for a structural computing environment [18]. The FOHM model forms a layer that is both general enough to serve multiple domains and specific enough to allow the particular features of these domains to be expressed fully. Most importantly, by identifying the commonalities of the domains, it minimizes the performance cost normally associated with such a generic approach.

REFERENCES

1. Anderson, K. M., Taylor, R. N., and Whitehead, E. J. A critique of the open hypermedia protocol. *Journal of Digital Information (JoDI). Special Issue on Open Hypermedia Systems 1*, 2 (1997).

2. Bush, V. As we may think. *The Atlantic Monthly 176*, 1 (1945), 101–108.

3. Campbell, B., and Goodman, J. M. HAM: A General Purpose Hypertext Abstract Machine. *Communications of the ACM 31*, 7 (July 1988), 856–861.

4. Davis, H. C., Rizk, A., and Lewis, A. J. OHP: A draft proposal for a standard open hypermedia protocol. In *Proceedings of the 2nd Workshop on Open Hypermedia Systems, ACM Hypertext '96, Washington, D.C., March 16-20. Available as Report No. ICS-TR-96-10 from the Dept. of Information and Computer Science, University of California, Irvine* (1996), U. K. Wiil and S. Demeyer, Eds., pp. 27–53.

5. Dervin, B. Given a context by any other name: Methodological tools for taming the unruly beast. In *Information Seeking in Context* (1997), Taylor Graham, London, pp. 13–38.

6. Engelbart, D. C. Augmenting human intellect: A conceptual framework. Tech. Rep. AFOSR-3223, Contract AF 49(638)-1024, Stanford Research Institute Technical Report, Palo Alto, CA, Oct. 1962.

7. Furuta, R., and Stotts, P. D. Programmable Browsing Semantics in Trellis. In *Proceedings of Hypertext'89* (Nov. 1989), pp. 27–42.

8. Halasz, F., and Schwartz, M. The dexter hypertext reference model. *Communications of the ACM 37*, 2 (1994), 30–39.

9. Lowe, D., and Hall, W. *Hypermedia & the Web. An Engineering Approach*. John Wiley & Sons, Chichester, 1999.

10. Marshall, C. C., and Shipman, F. M. Spatial hypertext: Designing for change. *Communications of the ACM 38* (1995), 88–97.

11. Marshall, C. C., and Shipman, F. M. Spatial hypertext and the practice of information triage. In *Proceedings of the '97 ACM Conference on Hypertext, April 6-11, 1997, Southampton, UK* (1997), pp. 124–133.

12. Moreau, L., and Hall, W. On the Expressiveness of Links in Hypertext Systems. *The Computer Journal 41*, 7 (1998), 459–473.

13. Nelson, T. H. Getting it out of our system. In *Information Retrieval: A Critical Review*, G. Schechter, Ed. Thompson Books, Washington, D.C., 1967, pp. 191–210.

14. Nelson, T. H. *Literary Machines*. Published by the author. Mindful Press, 1987.

15. Nürnberg, P. J. *HOSS: An Environment to Support Structural Computing*. PhD thesis, Department of Computer Science, Texas A&M University, College Station, TX, 1997.

16. Nürnberg, P. J., Grønbæk, K., Bucka-Lassen, D., Pedersen, C. A., and Reinert, O. A component-based open hypermedia approach to integrating structure services. *New Review of Hypermedia and Multimedia* (1999). Accepted for publication.

17. NÜRNBERG, P. J., AND LEGGETT, J. J. A vision for open hypermedia systems. *Journal of Digital Information (JoDI). Special Issue on Open Hypermedia Systems 1*, 2 (1997).

18. NÜRNBERG, P. J., LEGGETT, J. J., AND SCHNEIDER, E. R. As we should have thought. In *Proceedings of the '97 ACM Conference on Hypertext, April 6-11, 1997, Southampton, UK* (1997), pp. 96–101.

19. NÜRNBERG, P. J., SCHNEIDER, E. R., AND LEGGETT, J. J. Designing digital libraries for the hyper-literate age. *Journal of Universal Computer Scienc 2*, 9 (1996).

20. PARK, S. Structural Properties of Hypertext. In *Proceedings of the '98 ACM Conference on Hypertext, June 20-24, 1998, Pittsburgh, PA* (1998), pp. 180–187.

21. REICH, S., WIIL, U. K., NÜRNBERG, P. J., DAVIS, H. C., GRØNBÆK, K., ANDERSON, K. M., MILLARD, D. E., AND HAAKE, J. M. Addressing interoperability in open hypermedia: The design of the open hypermedia protocol. *New Review of Hypermedia and Multimedia* (1999). Accepted for publication.

22. REINERT, O., BUCKA-LASSEN, D., PEDERSEN, C. A., AND NÜRNBERG, P. J. CAOS: A collaborative and open spatial structure service component with incremental spatial parsing. In *Proceedings of the '99 ACM Conference on Hypertext, February 21-25, 1999, Darmstadt, Germany* (Feb. 1999), pp. 49–50.

23. STOTTS, P. D., AND FURUTA, R. Hypertext 2000: Databases or documents? *Electronic Publishing—Origination Dissemination and Design 4* (June 1991), 119–122.

24. THÜRING, M., HANNEMANN, J., AND HAAKE, J. M. Hypermedia and cognition: Designing for comprehension. *Communications of the ACM 38*, 8 (Aug. 1995), 57–66.

25. TOMPA, F. W. A Data Model for Fexible Hypertext Database Systems. *ACM Transactions of Information Systems 7*, 1 (Jan. 1989), 85–100.

26. TRIGG, R. A straw model for link traversal in open hypermedia systems. In *Proceedings of the 4th Workshop on Open Hypermedia Systems, ACM Hypertext '98 Conference, Pittsburgh, PA, June 20-24. Available as Report No. CS-98-01 from the Dept. of Computer Science, 6700 Aalborg University Esbjerg, Denmark* (1998), pp. 59–62.

27. VAN DYKE PARUNAK, H. Don't link me in: Set-based hypermedia for taxonomic reasoning. In *Proceedings of the '91 ACM Conference on Hypertext, Dec. 15-18, 1991, San Antonio, TX* (1991), pp. 233–242.

28. VAN DYKE PARUNAK, H. Hypercubes grow on trees (and other observations from the land of hypersets). In *Proceedings of the '93 ACM Conference on Hypertext, Nov. 14-18, 1993, Seattle, WA* (1993), pp. 73–81.

29. WIIL, U. K., AND NÜRNBERG, P. J. Evolving hypermedia middleware services: Lessons and observations. In *Proceedings of of the 1999 ACM Symposium on Applied Computing (SAC '99), San Antonio, TX* (Feb. 1999), pp. 427–436.

30. WIIL, U. K., AND ØSTERBYE, K., Eds. *Proceedings of the ECHT '94 Workshop on Open Hypermedia Systems* (1994). Technical Report R-94-2038, Dept. of Computer Science, Aalborg University.