A Comparison of Hyperstructures: Zzstructures, mSpaces, and Polyarchies

Michael J. McGuffin
Department of Computer Science
University of Toronto
Toronto, Canada
mjmcguff@cs.toronto.edu

m. c. schraefel
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom
mc@ecs.soton.ac.uk

ABSTRACT

Hypermedia applications tend to use simple representations for navigation: most commonly, nodes are organized within an unconstrained graph, and users are presented with embedded links or lists of links. Recently, new data structures have emerged which may serve as alternative models for both the organization, and presentation, of hypertextual nodes and links. In this paper, we consider zzstructures, mSpaces, and polyarchies from the perspective of graph theory, and compare these models formally. The novel aspects of this work include: providing a sound, graph-theoretic analysis of zzstructures; the identification of a new class of polyarchies associated with mSpaces; and the comparison and classification of these and other structures within a taxonomy. The taxonomy that results from our comparison allows us to consider, first; what the distinct characteristics of each model are at a fundamental level, and second; what model or attributes of a model may be most appropriate for the design goals of a given hypermedia application.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*navigation, theory*; E.1 [**Data Structures**]: graphs and networks; G.2.2 [**Discrete Mathematics**]: graph theory

General Terms

Human Factors, Theory

Keywords

Connective structures, ZigZag, zzstructures, mSpace, polyarchies, multitrees, edge-coloured graphs

HT'04, August 9–13, 2004, Santa Cruz, California, USA.

1. INTRODUCTION

In the era of the Web, hypertext has largely been experienced as the click and the page: single clicks on single anchors within single pages lead users to other single pages (that do not themselves link back to the invoking page). In hypertext research, this type of link, that points to but one instance, and works in only the outgoing direction, is a unary link. The unary link has defined interaction on the Web to be a perpetual message of "go somewhere else:" each click takes one from where they are currently to somewhere else. Similarly, the page as the smallest datum has meant that information can largely only be rendered as huge chunks, where each click loads a new page/chunk and replaces the previous chunk. In other words, in clicking on a link, the original context is frequently lost because a new complete page (or new location in that same page) loads in a browser window, replacing the previous page. Occasionally, clicks open new windows containing the new page, with the new window partially occluding the previous one. Similarly, some browsers let users open links into "tabbed" windows, where tabs allow users to move quickly between these pages. With both multiple-open and tabbed windows, however, the problem is the same: links point to individually huge chunks of information (pages) rather than to either (a) flexibly sized information chunks (as in Web-based approaches such as ConTexts [20] or Fluid Links [27]) or (b) multiple possible references (as supported by linkbases [1]). As such, with current Web-based presentation strategies, contextual information is occluded or lost from view. Once pushed into memory like this, cognitive load for constructing information context is increased.

Over the past few years, there has been work both in the hypertext and interaction communities that has considered how to represent smaller-than-page-sized amounts of data to deliver more flexible ways to interact with information, and to maintain context of that information. These models, developed independently of each other, include zzstructures[1] [18], mSpaces [21], and polyarchies [19]. In each case, the approach has been to use models which rely on the schema or semantics of the information in order to represent both the desired information itself, as well as the relations among the information. Further, these contextually associated pieces of information are themselves actively available to the user: the user can move immediately from their current focus of

[1]Some researchers prefer the form zzStructure, with a capital *s*; we, however, have followed the example in [14].

interest to the new focus of interest, without losing how that new information is related to the previous focus.

The effect of these approaches is that it is possible to generate richer models for interacting with information than with the single click-to-page approach. Each approach supports the representation of persistent context: one sees the node of interest in relation to other associated nodes. But more than this, one sees the node of interest from that node's position relative to multiple perspectives or orientations. For instance, in representing a Classical Music domain, one could look at Mozart from the perspective of Romantic composers, or from the perspective of music used in contemporary films, or recordings made on non-traditional instruments. By presenting both the information and the possibility of a variety of contexts, people have the opportunity to go beyond the current Web paradigm of accessing only the information itself. We can begin to access information from multiple perspectives and a variety of contexts.

While the deployment of these models relies on the real data being metatagged in such a way that it can be automatically associated, this is no longer an outrageous concept for the heterogeneous, distributed Web. Much local data is already generated on the Web from database schemas. Likewise, the Semantic Web effort [4] is chiefly involved with research into tools and approaches that will support the coordination and connexion of heterogeneous sources in semantically meaningful ways — ways that can be reasoned over to create associations of the type we are describing.

That said, we wish to step back from the implementation-level view of applying these approaches to the Web, to look at them from a graph-theoretic perspective. By considering zzstructures, mSpaces and polyarchies as graphs, we can investigate formal comparisons to find where these models overlap, and where they are distinct. In addition, our graph theoretic analysis of these structures has produced some valuable by-products. First, identifying the precise graph structure of zzstructures has provided new insights about zzstructures, facilitating a deeper understanding of their key properties. Second, analysis of mSpaces has uncovered a new class of polyarchies associated with mSpaces, with interesting properties. Finally, in comparing these and other structures, we present a taxonomy of models, with initial observations of their key differences. As a whole, this serves as a first step in aiding hypermedia designers assess which models best support their design goals, and which attributes of the models are most appropriate for their systems.

In the following sections, we review each model, and extend or rephrase their current descriptions where necessary to provide graph descriptions of each. From these descriptions, we present a taxonomy of the models. Our research goal beyond this work is that, through this taxonomy, we can begin to build heuristics for selection, so that hypertext designers can determine which model might best support the representation and interaction aims of their systems.

2. BACKGROUND

The most familiar data structures for hypermedia applications, hypermedia navigation, and organization of information in general, are 1D arrays (i.e. lists), 2D arrays (e.g. spreadsheets or tables), hierarchical trees, and graphs. The constraints imposed to varying degrees by each structure have advantages and disadvantages: constraints help simplify user interfaces, but also limit a user's freedom. A 1D list can be navigated with a very simple interface, but may only be useful if the data to be organized is very small in number (such as a navigation list for a website), or has only one natural ordering (such as an alphabetic list in a dictionary). The value of this organization breaks down if the representation does not readily support the kind of access needed by the user. An online dictionary, for instance, with an alphabetic ordering of words, may be difficult to search if only the phonetic pronunciation of a word is known.

As another example, trees are widely used, for example, to organize file systems, sections of documents, and web site maps. A common problem encountered by users forced to use trees is difficulty in adding a new object, or finding an existing object, that may have multiple, equally valid locations in the tree.

One alternative to imposing any connective structure on a data set is to use a "flat" database. Each data item can be given a set of keywords or attributes which can then be searched on. A disadvantage of using this approach alone is that users cannot exploit their spatial memory and learn *where* things are located [2, 5], or remember paths of links to find them. Furthermore, there is no reason that attribute-based search cannot be combined with an underlying organizing structure. For example, at least one operating system supports the addition of arbitrary, searchable, user-defined attributes within a hierarchical file system [10]. Thus, whatever promise a searchable database approach may hold, there is still value in seeking a useful connective structure to further organize data items.

To address some of the shortcomings of lists, 2D arrays, and trees, we may look to graph theory for alternative structures. General graphs are very flexible, but without constraints, they can be unwieldy: they usually cannot be drawn without edges crossing, and if large and dense enough, may appear to the user as a tangled, uninformative mess. However, there may exist variations on or extensions to graphs, that impose useful constraints, while affording more flexibility than trees. For example, hypergraphs (see Cormen et al. [7] for a description of these and other kinds of graphs, and for a review of some of the standard graph theory terminology used in this paper) generalize links to be between more than 2 nodes, and can be compared to hypermedia systems that support multi-pointing links, as well as set-based hypermedia [25]. It is also interesting to consider connective structures that have an inherently dynamic structure, as with HTL* [24], though in this paper, we focus more on properties of static structures. Note that static structures could nevertheless be snapshots of a dynamic process.

Other structures have been introduced more recently. Multitrees [9] are a kind of directed acyclic graph (DAG) that can contain multiple overlapping trees that share subtrees. Multitrees can be used to encode multiple alternative hierarchies, and are also suitable for modelling human genealogies ("family trees"). Each node in a multitree has a tree of ancestors and a tree of descendants (Figure 1), and this serves as the basis for the *centrifugal view* visualization proposed by Furnas and Zacks [9].

Polyarchies [19] are another structure which can contain multiple overlapping trees. Unlike multitrees, where trees may only share subtrees, the trees in a polyarchy may overlap or intersect at arbitrary nodes. Furthermore, whereas the trees in a multitree are implied by the structure of the

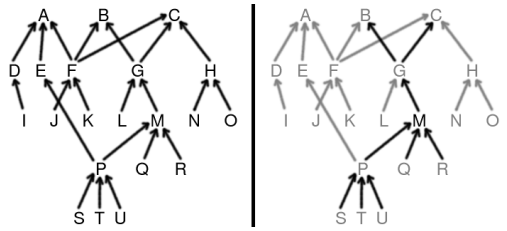

Figure 1: Left: an example multitree. Right: Node M is highlighted, along with its tree of ancestors and its tree of descendants.

multitree, in a polyarchy data structure, some extra information is required to somehow mark or identify each tree. Mathematically, one way to model this extra information is by colouring edges to identify different trees. A polyarchy can thus be thought of as an edge-coloured multigraph (Figure 2). Robertson et al. [19] developed a technique for smoothly transitioning from one tree in a polyarchy to another, by rotating the trees in 3D around a *pivot point*, i.e. a node where the two trees intersect.

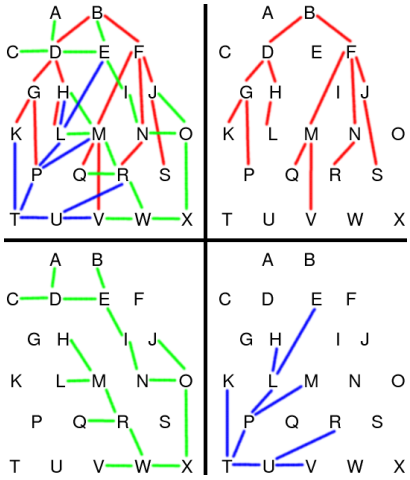

Figure 2: Top left: an example polyarchy of 3 trees, each marked with a different edge colour. The other 3 images show each constituent tree.

Zzstructures [14, 16, 18] is the generic name for Ted Nelson's ZigZag®, which generalizes (i.e. subsumes) lists, 2D arrays, and trees. Zzstructures also allow multiple instances of lists, 2D arrays, and trees to coexist within the same data set, each organizing the data in a different way. Applications proposed for zzstructures include hypermedia [17], personal schedules [16], and programming [15]. Previous introductions [14, 16, 12] to zzstructures emphasize a non-Euclidean spatial mental model, based on *dimensions* along which connections are made. In our experience, this has tended to make zzstructures difficult for novices to understand, and also makes it difficult to compare the formal structure of zzstructures to other connective, graph-like structures. In Section 3, we use standard graph-theoretic concepts to define a special kind of graph equivalent to zzstructures. This enables a better understanding of zzstructures, and eases comparison with other kinds of graphs.

mSpace [21, 11] is an approach for visualizing and interacting with multidimensional or multivariate data (e.g. the type of data found in a tabular database, where each data point, corresponding to a row in the database, has the same number of attributes, each corresponding to a column in the database). The interaction in mSpace combines "preview cues, dimensional sorting and spatial context" and is geared toward "user-determined adaptable content" [21]. In Section 4, we show that the structure exposed for navigation by an mSpace interface (e.g. Figure 3) is a special kind of polyarchy.

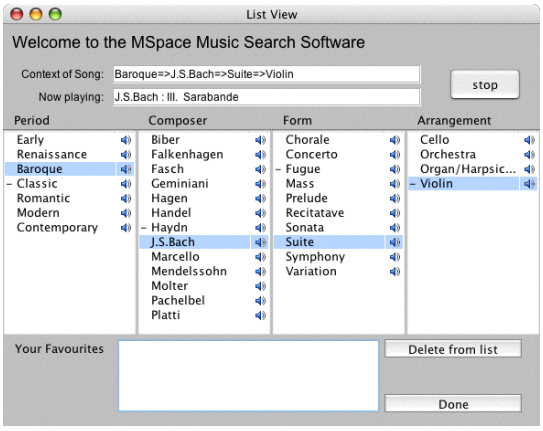

Figure 3: An mSpace multipane browser, viewing a database of classical music. The columns, from left to right, are *Period*, *Composer*, *Form*, and *Arrangement*. Each column corresponds to a level in a tree of choices. The children of the root node are in the left-most column, and leaves are in the right-most column. Selecting an item in every column corresponds to specifying a path through the tree. The left-to-right ordering of columns may also be changed, changing the structure of the tree, thereby allowing many possible trees to be browsed.

3. ANALYSIS OF ZZSTRUCTURES

Zzstructures can be formulated according to at least three different points of view, or interpretations: graph-oriented, list-oriented, and space-oriented. We begin with a graph-oriented approach, and *define* zzstructures as a kind of graph, because this approach allows for a simple and unambiguous definition of zzstructures, and can be easily understood by readers familiar with graph terminology. Relationships with the other two approaches are then identified.

We define zzstructures as a kind of *directed multigraph* — by *directed*, we mean that the edges of the graph have a direction associated with them, as shown by arrows in depictions of the graph; by *multi*graph, we mean that each pair of nodes may have multiple edges between them. Specifically, a zzstructure is a directed multigraph, with coloured edges, satisfying the following restriction:

> Restriction R: each node in a zzstructure may have at most one incoming edge of each colour, and at most one outgoing edge of each colour.

The effect of restriction R is that the edges of each colour form paths and/or cycles that do not intersect within the same colour (Figure 4).

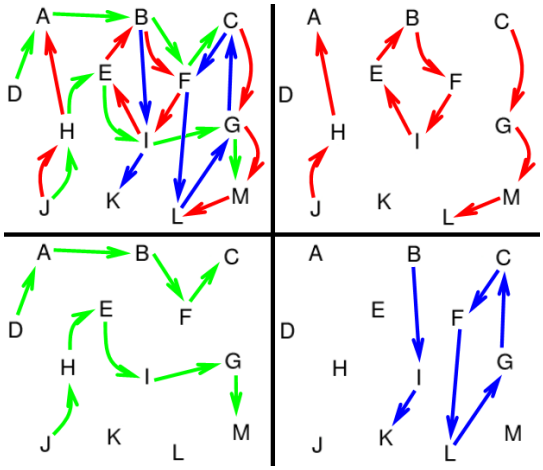

Figure 4: Top left: an example zzstructure involving 3 edge colours. The other 3 images show the edge subsets of each colour. Note how each subset of edges is a set of non-intersecting paths and/or cycles.

According to the list-oriented approach [12], a zzstructure is a set of nodes together with a set of *lists* of nodes. Each list has an associated (not necessarily unique) colour, and each node may appear in at most one list of each colour. Thus, each list gives the nodes forming a path or cycle whose edges are of the list's colour. The zzstructure in Figure 4 could be described with the following lists, where each list is prefixed with its colour: (red,(J,H,A)), (red,(B,F,I,E,B)), (red,(C,G,M,L)), (green,(D,A,B,F,C)), (green,(J, H,E,I,G,M)), (blue,(B,I,K)), (blue,(C,F,L,G,C)). Each list corresponds to what Nelson calls a *rank* of cells [14]. (Interestingly, we could have defined zzstructures as a kind of hypergraph [7], where each hyperedge is an ordered tuple corresponding to one list (or rank), and hyperedges of the same colour may not intersect.) The list-oriented approach emphasizes *paths* through the structure, which is relevant within the context of path-centric browsing research [8].

To understand the more usual space-oriented point of view, let N be the number of edge colours in a zzstructure Z. Imagine embedding Z in an N-dimensional space, where each of the edge colours corresponds to a different spatial *dimension*. For example, the 3 edge colours red, green, and blue in Figure 4 can be identified with 3 spatial dimensions x, y, and z respectively. The green edge connecting nodes A to B corresponds to a connection along the y spatial dimension. Thus, travelling from a node to an adjacent node along an edge of colour C corresponds to moving through the space along C's dimension; furthermore, ranks of the same colour are parallel. The space containing Z, however, is not necessarily Euclidean: edges in a zzstructure can act as "wormholes" that "teleport" the user to arbitrary, seemingly distant, nodes. For example, in Figure 4, starting at node B, we may travel a distance of one edge along the positive blue (z) direction to arrive at node I, or travel a distance of two edges along the positive red (x) direction (which is orthogonal to the blue direction) and arrive at the *same* node I! It is in this sense that the space associated with a zzstructure is non-Euclidean.

Returning now to the graph-oriented point of view of zzstructures, some remarks can be made regarding our definition of zzstructures. The essential role played by edge colours in our definition is providing a way of "typing" the edges. In an alternative definition, labelling each edge with an integer from 1 to N would do just as well. Secondly, the reason the edges of a zzstructure are directed is so we may differentiate between what Nelson calls *posward* and *negward* [14] directions — i.e., with or against an edge's direction. As per graph theory conventions, directed edges are depicted with one-way arrows in our figures. However, we do not suggest that the links in a zzstructure may only be traversed in one direction, nor that the zzstructure should be implemented with one-way pointers. Quite the contrary: links in zzstructures should be traversable in either direction.

Visualizing zzstructures is challenging, since the space associated with a zzstructure may have a large number of dimensions and be non-Euclidean. Drawing such a structure can result in multiple edge crossings, and/or inability to straighten out or flatten dimensions, and/or occlusion problems, depending on the drawing strategy employed. One traditional approach to visualizing zzstructures is a 2D cursor-centric view [14, 16] where a subset of nodes, arranged along 2 dimensions and centred at a cursor, is extracted and displayed in a flat, 2D view. To make it easy to flatten the subset, it must match one of the templates in Figure 5: a column view, or H-view, so named because the columns are reminiscent of the vertical bars in a capital letter H; or a row view, or I-view, with rows reminiscent of the horizontal serifs in a capital letter I. These templates extract ranks (i.e. "columns" or "rows") along the 2 chosen dimensions (Figure 6). The non-Euclidean nature of the space is dealt with by displaying virtual copies of nodes that have multiple apparent locations with respect to the cursor (e.g. Figure 6, top right: from A, we can arrive at C by travelling right two units, or up one unit).

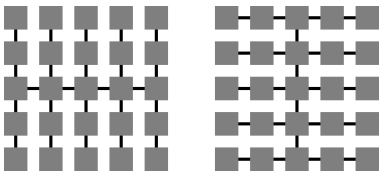

Figure 5: Templates of nodes that can be extracted from a zzstructure and easily flattened for viewing. In both cases, the cursor is located at the centre node. Left: an H-view. Right: an I-view.

Within a 2D cursor-centric view, three actions are available to the user for navigation, and each may hide, reveal, or rearrange nodes in a different way: changing the cursor position, changing templates (compare Figure 6 top right and bottom left), and changing the 2 dimensions used in the 2D view (compare Figure 6 bottom left and bottom right).

Zzstructures generalize, or subsume, lists and 2D arrays — Figure 7 shows how these can be realized as special cases of zzstructures. Although restriction R precludes use of one-to-many links within the same edge colour, there are "workarounds" that enable zzstructures to also generalize trees (Figure 8). Figure 9 shows how H- and I-views can be used to navigate a hierarchical menu. Additional examples of zzstructures are available online [14, 13].

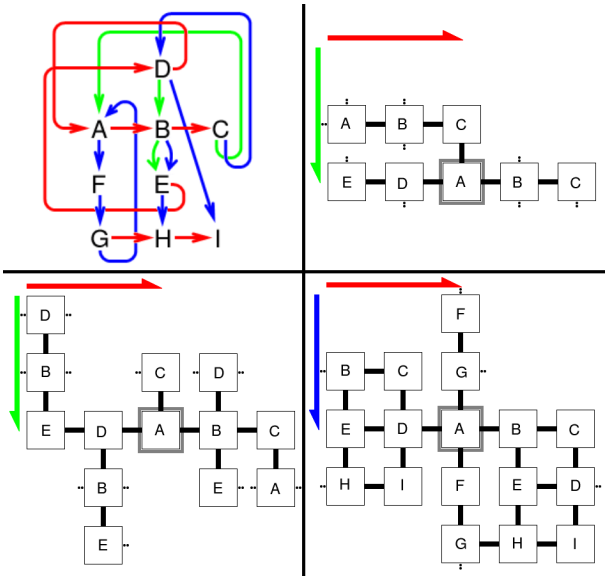

Figure 6: Top left: an example zzstructure involving 3 edge colours. Top right: an I-view with the cursor at A. Bottom left: an H-view with the cursor at A. Bottom right: same as bottom left, but with a different edge colour mapped to the vertical axis.

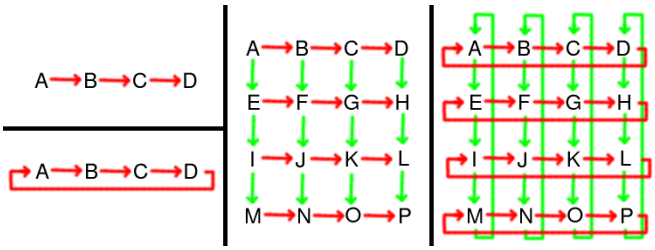

Figure 7: Example zzstructures. Upper left: a 1D array, or list. Lower left: a wrap-around list, or cycle. Middle: a 2D array, or spreadsheet, or table. Right: a wrap-around 2D array, or torus.

Not only can lists, 2D arrays, and trees be encoded in zzstructures, but even multiple instances of each of these may be encoded simultaneously, by using different edge colours for each instance. For example, one might organize nodes within a tree using two edge colours, and also organize them within a list using a third edge colour, and also organize them within an alternative tree using two more edge colours, all within a single zzstructure. Each sub-structure can be viewed by mapping the appropriate edge colours to the axes of the 2D cursor-centric view. Such combinations allow the user to have many alternative sub-structures available, each organizing and arranging the nodes in a different way.

4. ANALYSIS OF MSPACE

mSpace interfaces have the ability to organize the data points in a multivariate space as a tree, and allow this tree to be browsed, for example with a multipane browser (Figure 3). Furthermore, the levels of the tree, which each correspond to a dimension of the space, may be re-ordered (using *dimensional sorting* [11]), changing the structure of the tree

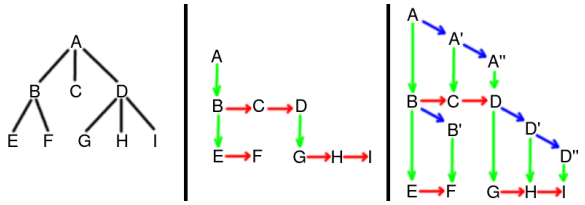

Figure 8: Left: a traditional tree, involving one-to-many links. Middle: a zzstructure encoding the same tree, without using explicit one-to-many links, but at the cost of using 2 edge colours. This encoding is analogous to the left-child, right-sibling pointer implementation of trees [7]. Right: an alternative encoding of the tree in a zzstructure, requiring 3 edge colours. Parent nodes are cloned along the 3rd edge colour.

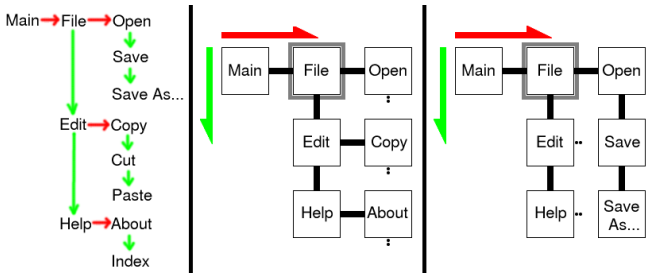

Figure 9: Left: a tree of menu commands, encoded in a zzstructure. Middle: an I-view with the cursor at "File" shows menu items at the same level as "File". Right: an H-view with the cursor at "File" shows menu items under "File".

and allowing many possible trees to be browsed. This is different from merely changing the ordering of items *within* a dimension, e.g. sorting composer names alphabetically by given name rather than by family name or by birth year.

We use the notation $\langle a_1, ..., a_L \rangle$ to denote an mSpace tree with L levels (not counting the root, which is at level 0), where $a_1, ..., a_L$ are the attributes corresponding to each level in the tree, from root to leaves (i.e. the left-most to right-most columns in a multipane browser). For example, the tree browsed in Figure 3 is the ⟨ *Period*, *Composer*, *Form*, *Arrangement* ⟩ tree. Dimensional sorting allows columns to be re-ordered, so that the user may, for example, swap (or *transpose* [11]) the first and second columns, revealing the ⟨ *Composer*, *Period*, *Form*, *Arrangement* ⟩ tree. This tree, unlike the former, makes it easy to see the period(s) corresponding to a selected composer, because *Composer* is the top-level choice.

As another example, Figure 10 shows the structure of the $\langle z, x, y \rangle$ tree for a small multivariate space. (Note that, for simplicity, the multivariate space in Figure 10 is shown fully populated with data points. In general, however, a multivariate space may only be sparsely populated.) Each dimensional sorting, or permutation of dimensions, corresponds to a different tree. With an L-variate space, there are $L!$ different trees, each of depth L, and each covering the data points with their leaf nodes. For example, Figure 10 shows one of $3! = 6$ possible trees, the other five being $\langle x, y, z \rangle$, $\langle x, z, y \rangle$, $\langle y, x, z \rangle$, $\langle y, z, x \rangle$, and $\langle z, y, x \rangle$.

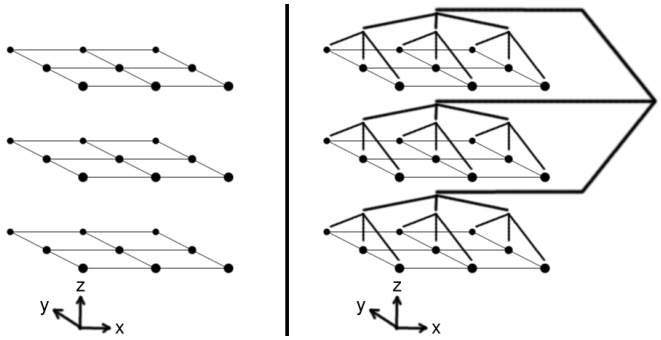

Figure 10: Left: an example multivariate space (of 3 dimensions), populated by $3 \times 3 \times 3$ data points. Right: one possible tree that covers the space. The three levels of the tree, from root to leaves, correspond to selecting values for z, x, and y, respectively. There are $3! = 6$ different permutations of the 3 dimensions, each corresponding to a different tree.

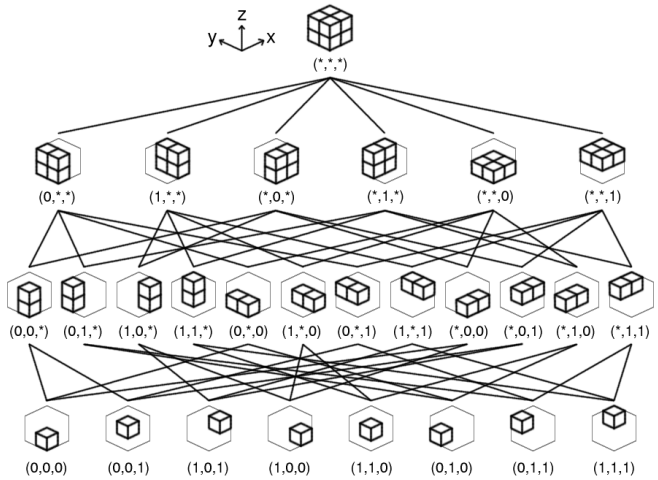

Figure 11: An mSpace polyarchy for a 3-dimensional, $2 \times 2 \times 2$ multivariate space. There are $3! = 6$ overlapping binary trees, each structured differently, but covering the same leaves (i.e. data points of the space). Each node corresponds to a slice of the space: the root is the entire 3D space, its children are 2D slices of the space, their children are 1D slices, and the leaves are zero-dimensional data points.

The $L!$ trees are alternative, overlapping trees, and thus constitute a polyarchy which we call an *mSpace polyarchy*. It is important to understand that mSpace polyarchies are in fact a special case of general polyarchies. An mSpace polyarchy contains all the possible mSpace trees for a given multivariate space, where the structure of each tree is constrained by the attributes (or dimensions) of the multivariate space, i.e. with one level per attribute. Removing a single tree from an mSpace polyarchy, or adding other trees that don't have a one-level-per-attribute structure, would still result in a polyarchy, but not an mSpace polyarchy. Thus, mSpace polyarchies are a specialized or constrained kind of polyarchy.

Figure 11 shows all the overlapping trees in the mSpace polyarchy of a tiny, $2 \times 2 \times 2$ space. Not only are leaves shared between the different trees, but subtrees are also shared in some cases (e.g. the trees $\langle x, y, z \rangle$ and $\langle y, x, z \rangle$ share the 4 subtrees under nodes $(0, 0, *)$, $(0, 1, *)$, $(1, 0, *)$, and $(1, 1, *)$). Unlike multitrees, however, where *only* subtrees may be shared between trees, the trees in an mSpace polyarchy can intersect at multiple, not necessarily contiguous, levels. For example, in a 7-variate space with dimensions $a_1, ..., a_7$, the trees $\langle\ a_1,a_2,a_3,a_4,a_5,a_6,a_7\ \rangle$ and $\langle\ a_2,a_1,a_5,a_3,a_4,a_7,a_6\ \rangle$ share all nodes on levels 0 (root), 2, 5, and 7 (leaves). (The nodes at a given level are shared between two trees whenever the sequence of attributes from the root to that level involves the same set of attributes in both trees. The root and leaves are always shared.)

One additional feature of mSpace interfaces not yet discussed is that they allow the user to limit their browsing to a *slice* of the full multivariate space. The location, orientation, and dimensionality of the slice currently browsed can be changed using various operations. *Expansion* operations add dimensions to the slice, *contraction* operations remove dimensions, and *substitution* operations replace a selected dimension with another [11]. Each of these operations creates a variant of the original slice, and each slice and its variants has an associated polyarchy. However, every slice's polyarchy is a sub-polyarchy of the full space's polyarchy. Furthermore, every slice corresponds to a node within the full space's polyarchy (Figure 11). So, without loss of generality, we limit our analysis here to the full space's polyarchy.

It is informative to consider what happens when a user transitions from one mSpace tree to another by transposing (i.e. swapping) two dimensions. Let $\langle a_1, ..., a_L \rangle$ be the tree before transition, and let i and j be such that a_i and a_j are the dimensions transposed, where $1 \leq i < j \leq L$. Then, after transposition, the tree is $\langle\ a_1,\ ...,\ a_{i-1},\ a_j,\ a_{i+1},\ ...,\ a_{j-1},\ a_i,\ a_{j+1},\ ...,\ a_L\ \rangle$. The last $L - j$ dimensions of both trees are $a_{j+1}, ..., a_L$, and on levels j through L the trees share all nodes. Thus, the two trees share all subtrees that are at depth j, corresponding to *at least* the leaf nodes when $j = L$. In addition, the first $i - 1$ dimensions are also the same in both trees, meaning they share nodes on levels 0 to $i - 1$, corresponding to at least the root node when $i = 1$. Thus, when transitioning between two trees in an mSpace polyarchy, there are many nodes in common between the two trees. This is in contrast to transitions between trees in a general polyarchy, where trees may only intersect at a single node.

Examining previous work, mSpace's multipane browser is not the only interface that makes the mSpace polyarchy available for navigation. FOCUS [23], later renamed InfoZoom, is a database browser designed for exploration, comparison, and dynamic queries within, for example, catalogues. Users may sort database entries by one attribute, then filter out entries that don't have a given value of the attribute, then sort remaining entries by another attribute, and repeat, successively drilling down to a smaller and smaller set of entries. By changing the sequence in which attributes are sorted on, users may effectively travel down any tree of the mSpace polyarchy. However, because FOCUS is motivated by different design goals, the tree-like nature of the data is not as apparent as in mSpace's multipane browser.

Interestingly, the connection between InfoZoom's interface and trees has only recently been made explicit [22].

Looking at other previous work, Conklin et al. [6] describe other specialized polyarchies. Of these, the *tuple polyarchy* is the most relevant to the current work. In tuple polyarchies, there is one tree for each dimension, where each dimension is broken down hierarchically (e.g. geographic dimension: continent, country, region, etc.). In the case of mSpace, however, the previous and current work on mSpace does not involve dimensions that are each broken down hierarchically in this fashion. Furthermore, in an mSpace polyarchy, each tree involves *all* the dimensions of the space.

5. COMPARISON OF HYPERSTRUCTURES

The two previous sections identified the underlying graph structures in zzstructures and mSpace. We may now compare these to polyarchies and other graph structures, and consider their respective properties.

5.1 Taxonomy

Figure 12 gives our taxonomy of structures, in the form of a subsumption diagram. A class of structures subsumes another class if the former is more general and can capture the latter as a special case.

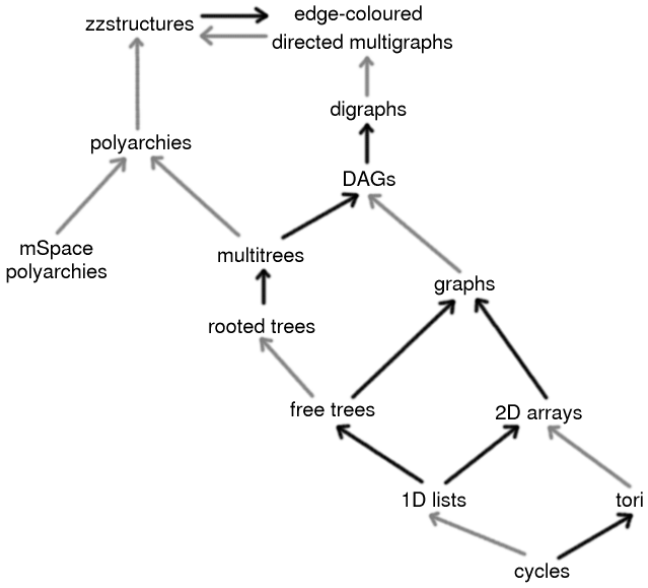

Figure 12: A subsumption diagram. Structures that appear higher in this diagram (i.e. that are at the receiving end of arrows) are more general than, and subsume, lower structures. Here, "graphs" refers to simple, undirected graphs. Black arrows show when a class of structures is a *subset* of the class pointed at (e.g. multitrees are a special kind of DAG). Grey arrows are used when the class is not technically a subset, but still subsumed "in spirit". For example, free trees are not a *kind* of rooted tree, hence the former is not a subset of the latter. However, any given free tree can be encoded as a rooted tree with no loss of information.

The relationships shown in Figure 12 depend in part on how each class of structures is defined precisely (e.g. are polyarchies defined as collections of intersecting *rooted* trees, or *free* trees?). In the interest of space, we do not list mathematically rigorous definitions for each class of structures to justify our diagram. Even without such a list, however, we feel that Figure 12 highlights important relationships that are based on fairly reasonable definitions.

Perhaps most interesting in Figure 12 are the two classes of structures at the top of the diagram, zzstructures and edge-coloured directed multigraphs, which subsume each other! Zzstructures are a subset (i.e. a kind) of edge-coloured directed multigraph, and are subject to restriction R, hence the black arrow from zzstructures to edge-coloured directed multigraphs. Although restriction R forbids multi-pointing (i.e. many-to-one or one-to-many) links within the same edge colour, such multi-pointing links can be "simulated" in a zzstructure through cloning (Figure 13). Thus, any given edge-coloured directed multigraph can be converted to a zzstructure with no loss of information (establishing an isomorphism between the nodes of the former and subsets of nodes of the latter), which explains the grey arrow pointing back to zzstructures in Figure 12. Although zzstructures are a strict subset of edge-coloured directed multigraphs, zzstructures are no less expressive.

Figure 13: Left: a node with multi-pointing links, forbidden by restriction R. Right: one way of simulating the same node in a zzstructure. Node A has been cloned once for each of its original edges, producing A, A′, A″, A‴, A⁗, and the clones are linked together with an edge colour not in the original structure.

One relationship was intentionally omitted from Figure 12. Technically, a digraph (directed graph) can always be encoded as a polyarchy, with no loss of information, by simply making every edge of the digraph a distinct, trivial tree in the polyarchy. However, such an encoding is not reflective of how digraphs and polyarchies are used in practice: it would be very cumbersome for a user to browse a digraph encoded in this way when using an interface designed for polyarchies of non-trivial trees. Hence, our taxonomy does not show polyarchies subsuming digraphs.

5.2 The Generality of Zzstructures

As explained toward the end of Section 3, zzstructures subsume lists, 2D arrays, and trees. Because zzstructures can encode multiple alternative sub-structures, they also subsume polyarchies. Finally, since multi-pointing links can be simulated in zzstructures, they also subsume *all* edge-coloured directed multigraphs — even those not subject to restriction R. Figure 12 shows zzstructures and edge-coloured directed multigraphs to be equivalent in terms of the information they can capture, and also positions them as the most general structures discussed.

Are zzstructures and edge-coloured directed multigraphs interchangeable? From the graph theoretic perspective we have given, restriction R is the *defining property* of zzstructures — the only thing differentiating them from edge-coloured

directed multigraphs. In fact, restriction R plays the same role as Nelson's two "Axioms of ZigZag®", namely, that "In any dimension, a cell can have at most one neighbor in each direction", and "In any dimension, a cell's positive side may only connect to a cell's negative side, and vice versa" [14]. Edge-coloured directed multigraphs have the same ability as zzstructures to encode multiple, overlapping sub-structures, and *also* allow true multi-pointing links within the same edge colour. One may wonder what value restriction R has, especially since, as shown by Figure 13, zzstructures can, in some sense, bypass it. It might seem that, at best, zzstructures and edge-coloured directed multigraphs are two different faces of the same essential concept, or at worst, that restriction R is an inconvenience, because it requires cloning of nodes to encode multi-pointing links.

We, however, feel restriction R *is* valuable in certain respects. Recall that restriction R requires each node have at most one incoming and one outgoing edge of each colour. As illustrated in Figure 4, this results in each colour corresponding to a set of non-intersecting paths and/or cycles. Furthermore, the absence of branching paths implies there is a unique way to move backward or forward along the path of each colour through a node. This means that the links in a zzstructure, though directed, are "2-way" links in some sense: if a user arrives at a given node from a given direction, returning to the previous node is simply a matter of moving back in the opposite direction. This would not be the case with multi-pointing links, where a user could arrive at the same node via the same "direction" (i.e. edge colour) from many different previous nodes. With restriction R, backing-up along an edge requires only remembering which colour was traversed. Without restriction R, backing-up requires remembering which node the user came from.

Another useful aspect of restriction R is that it can simplify viewing, and hence understanding, of a structure. Without restriction R, a node with multi-pointing links might have multiple neighbouring nodes to its immediate "right" and immediate "left". The 2D cursor-centric view for zzstructures would no longer apply — nor would a 3D extension of it that maps only one edge colour to each spatial dimension at a time. In fact, without restriction R, the whole spatial interpretation of zzstructures — of edge colours as dimensions — would be less workable.

Thus, there are definite ways in which the constraints imposed by restriction R can be valuable. At the same time, zzstructures can also circumvent restriction R (Figure 13) when needed, in which case it may be more appropriate to use a visualization interface designed to take cloned nodes into consideration.

5.3 Additional Comparisons

Figure 12 allows us to compare structures in terms of generality and differences in the kind of information captured by each. It also led us to think of additional ways in which structures may be compared, which we now describe.

Some structures can be compared in terms of the space associated with them. Many of the structures in Figure 12, such as graphs, trees, and polyarchies, have no particular space associated with them, although they could be *embedded* in one of many different spaces. Zzstructures and mSpace polyarchies, however, do have closely associated spaces that are, in some sense, intrinsic to the structures. The space usually associated with a zzstructure, and which we consider here, is the multi-dimensional, non-Euclidean space described in Section 3, where each edge colour corresponds to a spatial dimension. An mSpace's space, on the other hand, is the multivariate space containing data points, having one dimension for each attribute used to describe the data points. These two spaces differ in nature on many levels, which should help in determining which of them, if either, should be used in a given hypermedia application.

First, although both spaces can involve a high number of dimensions, a zzstructure's space is generally non-Euclidean, making it more difficult to flatten and visualize, hence the need for extraction of orderly subsets (Figure 5). In contrast, an mSpace's multivariate space is Euclidean, like a high-dimensional regular grid. This means that, for example, a 2D slice of the space can be taken, and shown on a 2D screen, without distortion.

Second, a zzstructure's space may be designed and changed independently of the content of the data nodes in the zzstructure. Given a set of nodes, the user may link them together in whatever zzstructure they please, whether manually, or by using algorithms to generate links, or some mixture of both. With mSpaces, however, the space is entirely determined by the choice of attributes used to describe each data point. Although the location of a data point within the multivariate space may be changed by changing the values of the point's attributes (i.e. its coordinates), and many different views of an mSpace are possible (e.g. by viewing different trees of the mSpace polyarchy), the multivariate space itself does not change.

Third, the nature of the dimensions are different in the two spaces. With an mSpace's multivariate space, each dimension, or attribute, corresponds to a discrete *variable*, such as a category (e.g. "period", "composer", "style"), or a quantity that is discrete, or that has been discretized through binning (e.g. "year", "cost"), or a string of characters (e.g. "name"). Furthermore, every data point has a *value* (such as "17th century", or "Beethoven") along *every* dimension (e.g. if "composer" is a dimension in the space, then every data point will have a composer attribute). In contrast, with zzstructures, dimensions are more often used as *containers* (e.g. "nodes of type x") than as variables, and nodes do not have an absolute value along each dimension, but rather have a location *relative* to their neighbours (e.g. "negward from node A"). Furthermore, a node in a zzstructure may only be connected along *some* dimensions, and have no connections along other dimensions. For example, if blue is an edge colour, there is no requirement that all nodes have a blue path through them.

This table provides a summary of our observations so far:

zzstructure's space	mSpace polyarchy's space
non-Euclidean	Euclidean
Space may be changed independently of content	Space determined by attributes on the content
Dimensions are like *containers*; Nodes have a *relative* position along *some* dimensions	Dimensions are *variables*; Data points have a *value* along *every* dimension

Having considered how the structure's spaces differ, another aspect to consider is the affordances of each structure with respect to transitioning between different views of the structures. Since each of these structures may, in general, be quite large, especially in hypermedia applications, users will often want to look at only a subset of the structure at

a given time. When transitioning between different subsets, or views, it is desirable to maintain context during transitions, for example by using smooth animations [26, 3], so that users do not become disoriented.

In particular, we consider transitions between different trees contained in the structures. Polyarchies, zzstructures, multitrees, and mSpace polyarchies all offer different ways of combining multiple trees into a single structure. When viewing an individual tree of any of these structures, the layout and navigation of the tree may be done following one of the many existing techniques for visualizing trees. However, when transitioning from one tree to another, it is not always clear how to best display the transition. Different kinds of overlap between the two trees suggest different approaches.

In the case of polyarchies, the two trees involved in a transition can intersect in arbitrary ways, and may only intersect at a single node. The visual pivot technique of Robertson et al. [19] uses 3D rotation and fading to gradually reveal a new tree while removing the original tree. Because both trees are temporarily visible together, the user has an opportunity to simultaneously see both the old context they are leaving, and the new context they are entering. However, the only node shown to be shared by the two trees is at the pivot point, and this is the only node that is visually preserved throughout the transition. Even if the two trees overlap at multiple nodes, the visual pivot technique does not show this. Furthermore, because the two trees can intersect in arbitrary ways, it is unobvious how to to design a transition that visually preserves all the shared nodes in a way that is easy to understand for the user.

Zzstructures suffer from the same problem just described for polyarchies. With zzstructures, the mechanism of transition between trees may be different (e.g. changing the mapping of axes in a 2D cursor-centric view from the dimensions of one tree to the dimensions of another), however, the issue is the same: the two trees can intersect in arbitrary ways, possibly only at a single node.

With multitrees, when moving from one tree to another, the trees may only overlap by sharing subtrees. Contrary to polyarchies, the overlap in this case is regular and predictable, and is easier to exploit during display of the transition. As the user moves from one tree to another, a smooth animation could "scroll" the user's view over the multitree, visually preserving the entire subtree shared by the old and new trees. In a centrifugal view [9], this can even be shown in two different directions — when moving from one focal node to another, overlap may occur between the descendant trees of the focal nodes, and also between the ancestor trees of the focal nodes.

With mSpace polyarchies, there is also predictable overlap between trees. As described toward the end of Section 4, when moving from one tree to another via a transposition of dimensions, the two trees share *all* subtrees at a certain depth, depending on the dimensions transposed. This presents interesting opportunities for visualization during the transition. How can we best display the multiple, shared subtrees during a transposition operation? In the multipane browser, this has not been a concern, because the multipane browser only displays one path through an mSpace tree at a time, rather than the entire tree. However, alternative visualizations may display larger portions of an mSpace tree, in which case the shared subtrees become an important consideration for designing smooth transitions.

Again, we summarize the foregoing comments with a table:

Structure	Nature of overlap between two trees in the same structure
polyarchy	arbitrary
zzstructure	arbitrary
multitree	one subtree is shared
mSpace polyarchy	all subtrees at a certain depth are shared

6. FUTURE DIRECTIONS

The structures discussed in this paper have been analyzed from a theoretical point of view. To learn more about these structures in practice, they need to be applied to more problems and more domains. Such applications would open the way to more meaningful, user-based evaluations of the models, and would also help in the development of practical heuristics for choosing one model over another.

Developing new visualization and interaction techniques for these structures could prove very valuable, e.g. for privileging structural attributes not exploited in existing techniques. This, however, remains challenging, due to the relatively sophisticated properties of the structures.

In particular, developing better visualizations for smooth transitions between substructures could help users maintain context during navigation. In the case of mSpace polyarchies, this may require considering the multiple, overlapping subtrees between trees, and how best to display this overlap during transitions.

Finally, there is still room to explore other kinds of hybrid or extended structures not yet identified. For example, future work on mSpace could combine the concepts of a tuple polyarchy [6], where each dimension is broken down hierarchically, and an mSpace polyarchy. Such a combination would at least require extending the multipane interface of Figure 3 to handle hierarchical dimensions.

7. CONCLUSIONS

By considering new models for representing information which go beyond generic organizing structures, we can consider equally new approaches for representing hypermedia information spaces. To better enable innovations in these directions, we have presented a comparison of three such models, from a perspective based in graph theory. This approach has yielded a new definition of zzstructures (specifically: an edge-coloured directed multigraph subject to restriction R), and identified a new class of polyarchies associated with mSpaces. It has also allowed us to compare some of the fundamental attributes of each model. This formal comparison is a first step toward better equipping hypermedia designers to choose how to conceptualize the domains they wish to represent. Future work should build on this comparison to develop heuristics for designers who wish to provide new ways to think about and access domains. Even at this stage, however, we hope the current formal comparison provides designers with a taxonomy of conceptual as well as functional models through which they can consider how they will engineer their hyperspaces.

8. ACKNOWLEDGEMENTS

Thanks to Theophanis Tsandilas, Michael Wu, Alex Hertel and Gonzalo Ramos at University of Toronto, to Leslie

Carr at University of Southampton, and to our reviewers, for valuable feedback. Thanks also to Bill Buxton for encouragement and support, and to Ted Nelson for some valuable comments during an earlier stage of this research. Early stages of the mSpace work have been funded by CITO and NSERC, Canada; ongoing work is supported by the AKT project, EPSRC, UK.

9. REFERENCES

[1] C. Bailey, S. R. El-Beltagy, and W. Hall. Link augmentation: A context-based approach to support adaptive hypermedia. In *Proceedings of 3rd Workshop on Adaptive Hypertext and Hypermedia*, pages 55–62, 2001.

[2] D. Barreau and B. Nardi. Finding and reminding: File organization from the desktop. *SIGCHI Bulletin*, 27(3):39–45, July 1995.

[3] L. Bartram. Can motion increase user interface bandwidth? In *Proc. IEEE Conference on Systems, Man and Cybernetics*, pages 1686–1692, 1997.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[5] C. Chen and M. Czerwinski. From latent semantics to spatial hypertext—an integrated approach. In *Proceedings of 9th ACM Conference on Hypertext and Hypermedia*, pages 77–86, 1998.

[6] N. Conklin, S. Prabhakar, and C. North. Multiple foci drill-down through tuple and attribute aggregation polyarchies in tabular data. In *Proceedings of IEEE Symposium on Information Visualization (InfoVis)*, pages 131–134, 2002.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. 1990.

[8] P. Dave, U. P. Karadkar, R. Furuta, L. Francisco-Revilla, F. Shipman, S. Dash, and Z. Dalal. Browsing intricately interconnected paths. In *Proceedings of 14th ACM Conference on Hypertext and Hypermedia*, pages 95–103, 2003.

[9] G. W. Furnas and J. Zacks. Multitrees: enriching and reusing hierarchical structure. In *Proceedings of ACM CHI 1994 Conference on Human Factors in Computing Systems*, pages 330–336, 1994.

[10] D. Giampaolo. *Practical File System Design with the Be File System*. Morgan Kaufmann, 1999.

[11] N. Gibbins, S. Harris, and m. c. schraefel. Applying mSpace interfaces to the semantic web, 2003. http://eprints.ecs.soton.ac.uk/archive/00008639/.

[12] T. Lukka. A gentle introduction to Ted Nelson's ZigZag structure. (An incomplete work in progress) http://www.nongnu.org/gzz/gi/gi.html.

[13] M. J. McGuffin. A graph-theoretic introduction to Ted Nelson's zzstructures, January 2004. http://www.dgp.toronto.edu/~mjmcguff/research/zigzag/.

[14] T. Nelson. ZigZag tutorial. 1999. http://xanadu.com/zigzag/tutorial/ZZwelcome.html.

[15] T. Nelson. ZZ cell programming. http://xanadu.com/zigzag/fw99/ZZcellProg.html.

[16] T. H. Nelson. What's on my mind. Invited talk at the first Wearable Computer Conference, 1998. http://www.xanadu.com.au/ted/zigzag/xybrap.html.

[17] T. H. Nelson. Xanalogical structure, needed now more than ever: parallel documents, deep links to content, deep versioning, and deep re-use. *ACM Computing Surveys*, 31(4es):33, December 1999.

[18] T. H. Nelson. Zigzag (tech briefing): Deeper cosmology, deeper documents. In *Proceedings of 12th ACM Conference on Hypertext and Hypermedia*, pages 261–262, 2001.

[19] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Polyarchy visualization: visualizing multiple intersecting hierarchies. In *Proceedings of ACM CHI 2002 Conference on Human Factors in Computing Systems*, pages 423–430, 2002.

[20] m. c. schraefel. ConTexts: Adaptable hypermedia. In *Proc. AH 2000 Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 369–374, 2000.

[21] m. c. schraefel, M. Karam, and S. Zhao. mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia. In *Proceedings of AH2003 Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 217–234, 2003.

[22] M. Spenke and C. Beilken. Visualization of trees as highly compressed tables with InfoZoom, 2003. Unpublished entry in InfoVis 2003 Contest, held at IEEE Symposium on Information Visualization. http://www.cs.umd.edu/hcil/InfovisRepository/.

[23] M. Spenke, C. Beilken, and T. Berlage. FOCUS: the interactive table for product comparison and selection. In *Proceedings of ACM Symposium on User Interface Software and Technology*, pages 41–50, 1996.

[24] P. D. Stotts, R. Furuta, and C. R. Cabarrus. Hyperdocuments as automata: verification of trace-based browsing properties by model checking. *ACM Transactions on Information Systems (TOIS)*, 16(1):1–30, January 1998.

[25] H. Van Dyke Parunak. Don't link me in: Set based hypermedia for taxonomic reasoning. In *Proc. 3rd ACM Conference on Hypertext*, pages 233–242, 1991.

[26] D. D. Woods. Visual momentum: a concept to improve the cognitive coupling of person and computer. *International Journal of Man-Machine Studies*, 21:229–244, 1984.

[27] P. T. Zellweger, B.-W. Chang, and J. D. Mackinlay. Fluid links for informed and incremental link transitions. In *Proceedings of 9th ACM Conference on Hypertext and Hypermedia*, pages 50–57, 1998.