Navigating Terminology Hierarchies to Access a Digital Library of Medical Images *

Flip Korn† and Ben Shneiderman‡
University of Maryland
College Park, MD 20742

Abstract

Browsing is an interactive and exploratory process for finding information in a digital library that has advantages over search term queries in many situations. Some browsers display a concept space as a node-link graph diagram, but this can look chaotic for a graph of moderate complexity. The approach taken in this paper is to suppress some of the interrelationships (links) and order the concept space as a tree by some 'natural' hierarchy. The user can then explore hidden interrelationships by dynamically interacting with the system. We demonstrate the usefulness of browsing a hierarchy via this method in a prototype called MeSHBROWSE, a system for browsing terms from the NLM Medical Subject Headings tree. It displays a node-link tree diagram of the concept space and reveals hidden interrelationships when a node is clicked on by triggering related nodes scattered about the tree to become highlighted. In this paper we describe MeSHBROWSE, discuss semantic and algorithmic issues involved with it, and discuss its implications for further research.

Keywords: browsing, graph visualization, digital library

Introduction

Browsing is an interactive and exploratory process for finding information in a digital library that has advantages over search term queries because in many situations because it relies on recognition rather than recall. Often, a user is interested in browsing a large network of information in the form of keywords, referred to here as a *concept space*, in order to discover interrelationships. Some browsers display a concept space as a node-link graph diagram, but for any substantially complex system this results in a tangled web that is chaotic to the user because of edge crossings and a confusing placement of nodes. "For richly interconnected material or documents of a reasonable size and complexity, it is not possible to include everything in a single browser without the problem of presenting 'visual spaghetti' to the user." [Dil90]

The strategy taken in this paper is to not represent all of the interrelationships between concepts at once. Of course, simply suppressing some of the interrelationships of a concept space in order to create a tractable structure for browsing will oversimplify the space, stripping it of some of its most interesting and important interrelationships. Instead, the idea is to (visually) order the concept space according to some 'natural' and intuitive (hierarchical) taxonomy and then allow the user to explore the hidden interrelationships by dynamically interacting with the system. Nodes of the same hidden relationship are likely to be scattered about different locations of the tree, but they can be seen as an interrelated group by triggering the scattered nodes to be color-coded. We believe that this can be

*This research was supported by the National Library of Medicine Fellowship Program. Support was administered by the Oak Ridge Institute for Science and Education.

†Computer Science Department and Institute for Systems Research. Email: flip@cs.umd.edu

‡Computer Science Department, Institute for Systems Research, Center For Automation Research, and Human-Computer Interaction Lab. Email: ben@cs.umd.edu

useful in assisting the end-user to browse the contents of a digitial library. While this method is applicable to a digital library storing information from any domain, in this paper we focus on browsing a medical database. For this we have developed a prototype system called MeSHBROWSE.

The concept space used in MeSHBROWSE is comprised of portions of a National Library of Medicine publication called the Medical Subject Headings (MeSH) tree. The MeSH tree, consisting of approximately 18,000 total terms and going seven levels deep, contains common medical terminology pertaining to anatomy, histology, pathology, and more [Nat95]. In addition to being published annually, it is available over the Internet and can be downloaded from ftp://nlmpubs.nlm.nih.gov. MeSH logically partitions medical terminology into mutually distinct groups such as body regions, systems and tissue types and then hierarchically orders within these groups via aggregation and generalization. However, this grouping and hierarchical ordering necessarily fails to capture some interesting interrelationships between terms. For instance, it cannot reveal which anatomical structures are spatially located near each other within the body. Such an interrelationship is precisely the kind that would be left for the user to discover through dynamic exploration.

In this paper we describe MeSHBROWSE, a prototype that displays portions of the MeSH tree and allows the user to interactively explore hidden interrelationships. The MeSH is displayed as a node-link tree. Clicking on a node will trigger a highlighting "chain reaction" whereby any nodes currently on the screen that are interrelated (by some specified relationship) will be color-coded. The highlighting provides instantaneous feedback, and therefore requires an efficient algorithm for large trees. We will discuss the issues involved with this algorithm later on in this paper.

Among other things, MeSHBROWSE, in conjuction with MeSH as a controlled vocabulary, can be used as a front-end to the National Library of Medicine's Visible Human Project. The NLM is in the process of creating a large dataset of anatomical images of a male and female subject. This database includes MRI, CT, and cryosections. These images will be available to a large community of users of varying background and expertise. Because these datasets will be large (at least 15 Gigabytes) and could take weeks to download, it is important for users to retrieve only their desired slices. By clicking on nodes to formuate visual queries, MeSHBROWSE can be used to aid users in selecting and retrieving the precise anatomical structures of interest to them. The following section describes how the system works.

MeSHBROWSE

Usage

A window will come up that initially displays only the top-level node of the tree specified in *dotfile*. It is labeled "MeSH" and is colored green, the default color. Similar to an outline widget, the user can expand and contract individual nodes of the hierarchy by moving the mouse over to an unexpanded node and pressing the return key. This will result in the node's children appearing to the right of it, connected by horizontal edges (see Figure 1). The color of the new nodes will also be green by default. Pressing return over an already-expanded node will start a contraction, recursively deleting all descendants of that node and their respective edges. Expansion and contraction is useful when the MeSH hierarchy specified in *dotfile* is too large to be viewed on the screen at once because it allows for the display of only the regions of interest.

Initially MeSHBROWSE does not have any data of any interrelationships. They must be loaded into the system during runtime. Files storing interrelationships are in **.data** format and are stored as matrices (see next section for a description of the data structures). To load in a data file of interrelationships, hold down the right mouse button and select "load matrix" from the menu that appears. A dialog box will

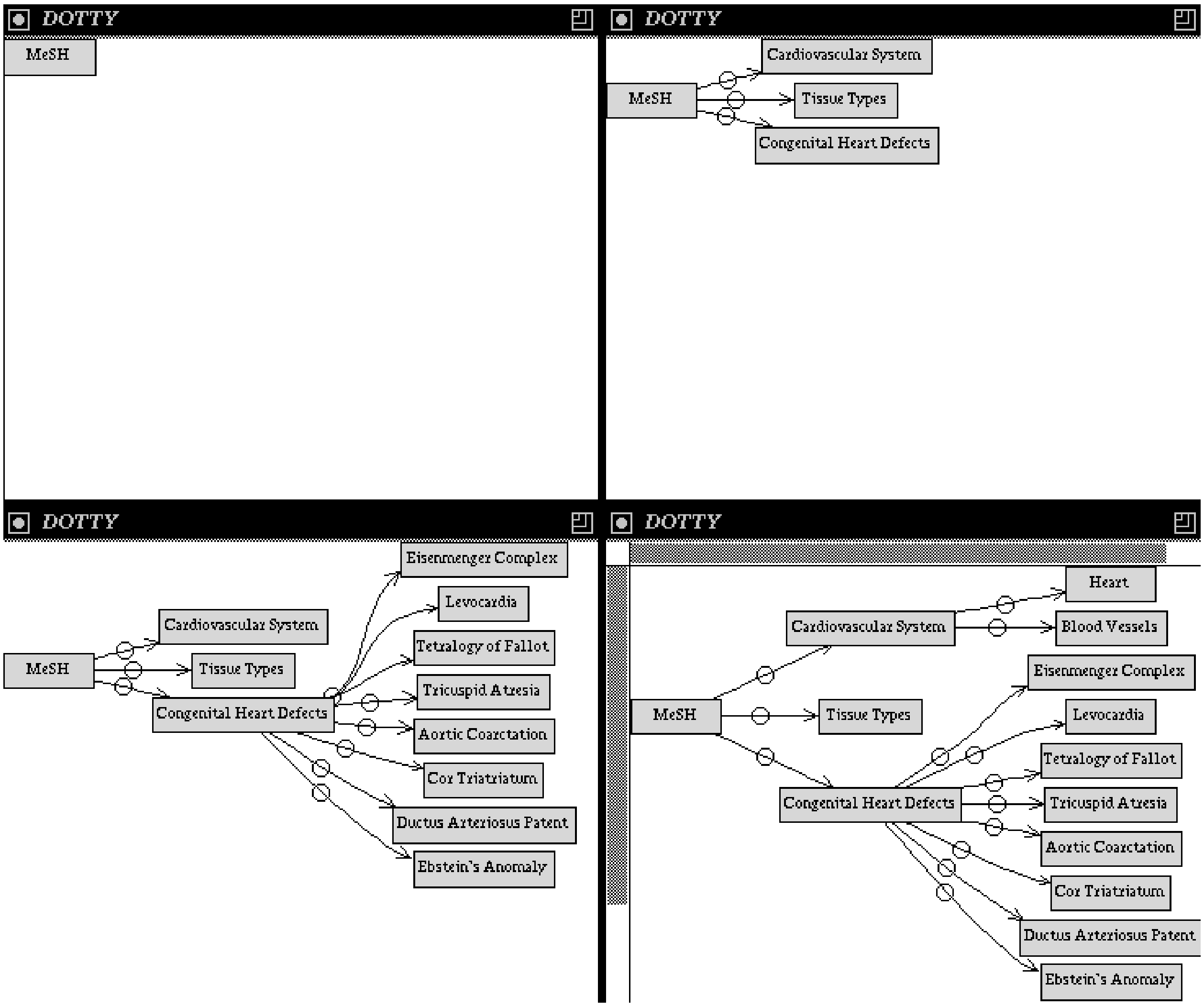

Figure 1: (a) Initial screen. (b) After expanding "MeSH". (c) After expanding "Congenital Heart Defects". (d) After expanding "Cardiovascular System".

then pop up for inputting the name of the data file. Only one data file matrix can be loaded at a time. Any subsequent data file matrices that are loaded will overwrite the current matrix in memory.

Once a matrix has been loaded, the system can respond to left mouse button clicks. Clicking on the left button over a node will activate an interrelationship "chain reaction" for that node if one exists (has been programmed into the loaded matrix). The node that has been clicked on will immediately change colors to yellow; all nodes that are interrelated (by whatever relationship the loaded matrix is supposed to signify) will be instantaneously colored red during what we are calling here the chain reaction. A Boolean AND query can be performed by clicking on another node, thus triggering another chain reaction. This time only those nodes that are both related to the newly-clicked on node *and* already colored red (from a previous chain reaction) will remain red. Those nodes that previously were red but are not related to the newly-clicked on node will return to their default green color.

To begin a new query, hold down the right mouse button and select "clear" from the menu. The results of any previous clicks (which together form a query) will be erased as the nodes return to their default green color. New queries can now be performed as if the data file matrix has just been reloaded.

Once a sufficient number of interrelationships has been explored to precisely determine the anatomical structures of interest, cryosection images from the Visible Human dataset can be downloaded by clicking with the middle mouse button on the node(s) labeled with the anatomical structures of interest.

An Example

The sample file *heart.dot* that comes with MeSHBROWSE includes the specification for a tree comprised of MeSH portions of the cardiovascular system, tissue types, and congenital heart defects. Figure 2 displays the fully-expanded tree. When we first run *dotty* on *heart.dot*, initially only the node labeled "MeSH" will be displayed. We can expand the tree to make the items on the second level of the tree appear: "Cardiovascular System", "Tissue Types", and "Congenital Heart Defects". Suppose we would like to learn more about a congenital heart disease called Teralogy of Fallot. Expanding "Congenital Heart Defects," we get a list of diseases of which "Teralogy of Fallot" is one of them. Specifically, we would like to know which anatomical structures are affected by Teralogy of Fallot, so we expand "Cardiovascular System" to get "Heart" and "Blood Vessels". We continue to expand the branches with these structures, opening up "Heart" and then "Heart Valves". Next we load in the interrelationship data by holding down the right mouse button, selecting "load matrix", and entering "heart.data" at the prompt. Now if we click on "Teralogy of Fallot" the node containing "Teralogy of Fallot" will be colored yellow and the nodes "Heart Septum", "Heart Ventricle", and "Pulmonary Valve" will be colored red (see Figure 3). We can interpret this as meaning that Teralogy of Fallot affects these structures. Now if we click on "Levocardia" we discover that Teralogy of Fallot and Levocardia affect different structures.

Now suppose we would like to find out which congenital heart defects affect the heart atrium. After clicking on "Heart Atrium" the nodes labeled "Levocardia", "Tricuspid Atresia" and "Cor Triatriatum" are triggered (see Figure 4). Finally, clicking on the "Heart Atrium" node with the middle button brings up a browser to the Visible Human dataset, developed by Chris North of the Human-Computer Interaction Lab at the University of Maryland, for retrieving cryosections of the heart atrium [NSP95]. The browser displays the range of slices involved and one characteristic slice (see Figure 5).

Creating Interrelationships

The MeSHBROWSE system comes with an editing tool to create new interrelationship groups and store them as a data file matrix. Existing data file matrices can also be modified by loading them, editing new interrelationships, clear-

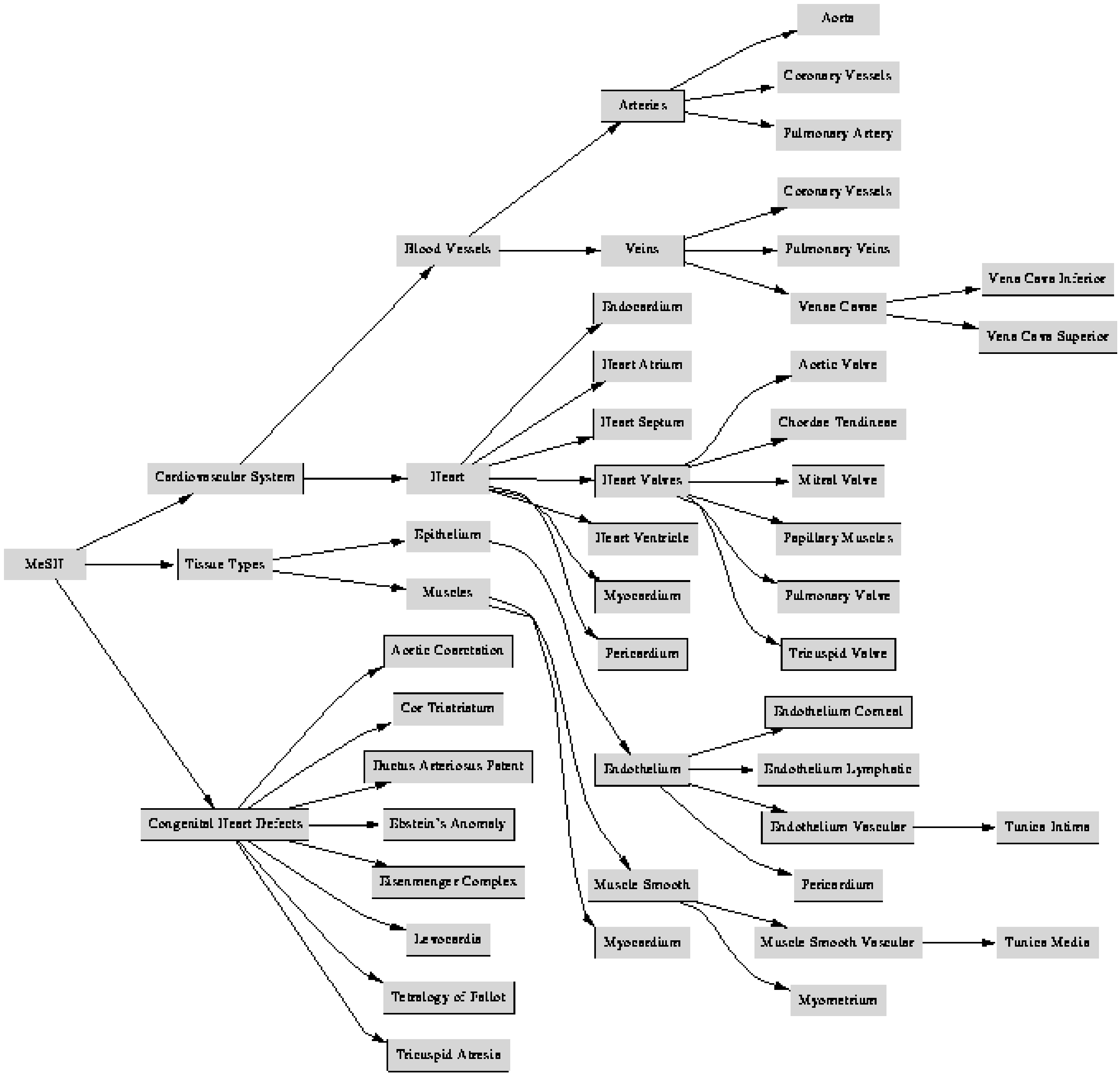

Figure 2: Congenital heart diseases tree created from portions of MeSH with subtrees for the cardiovascular system, cardiovascular tissue types, and congenital heart defects

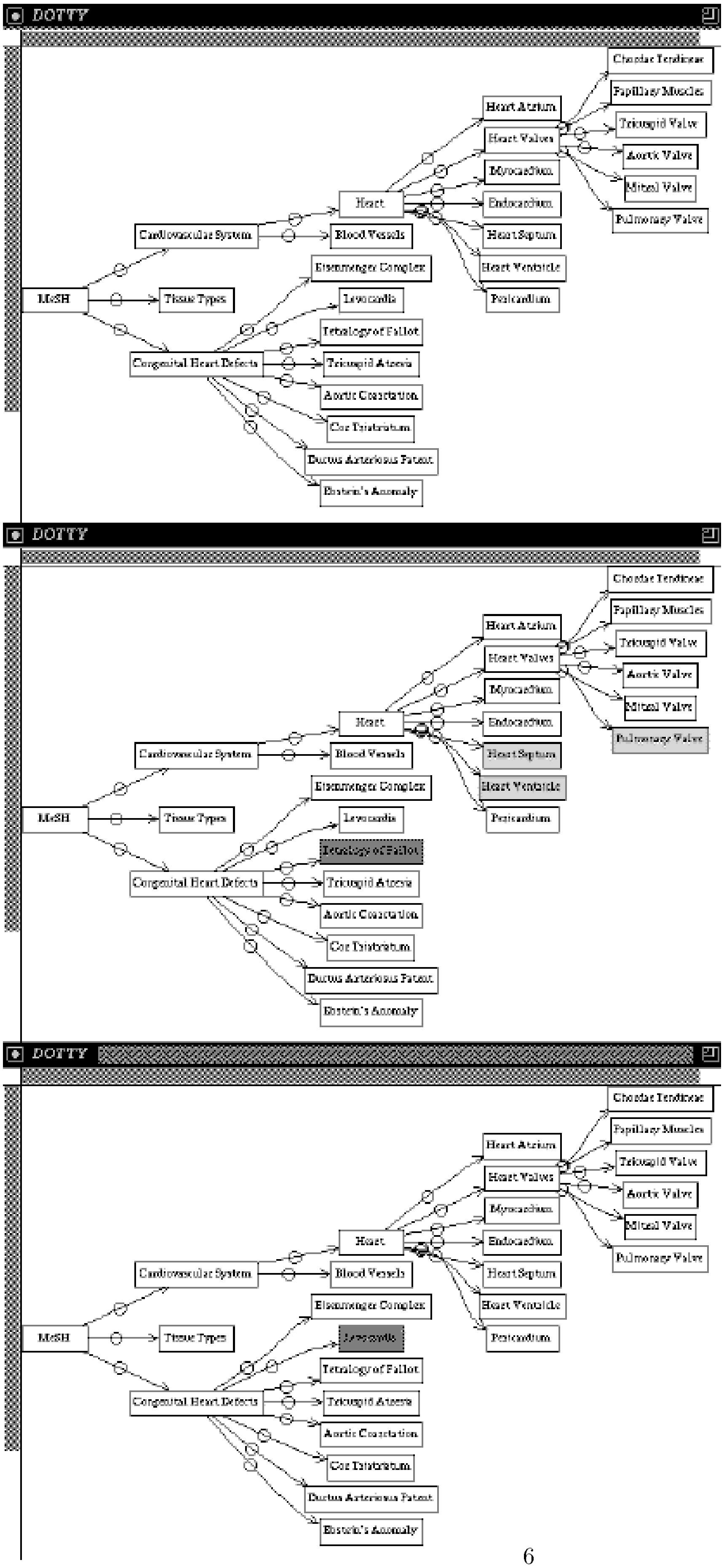

Figure 3: (a)Before query (b)After clicking on "Teralogy of Fallot" (c)ANDed with "Levocardia"

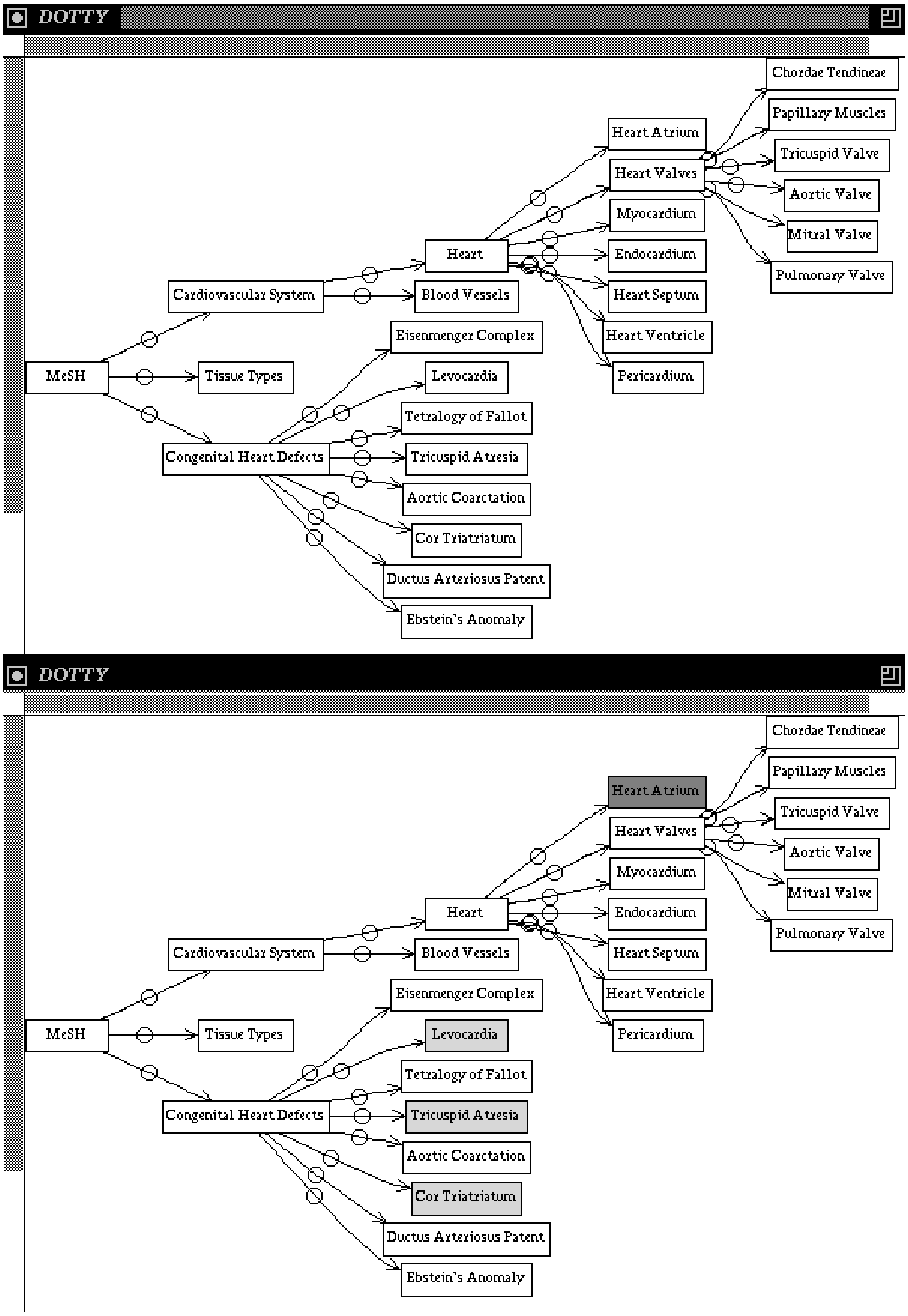

Figure 4: Clicking on "Heart Atrium" triggers "Levocardia", "Tricuspid Atresia", and "Cor Triatriatum".

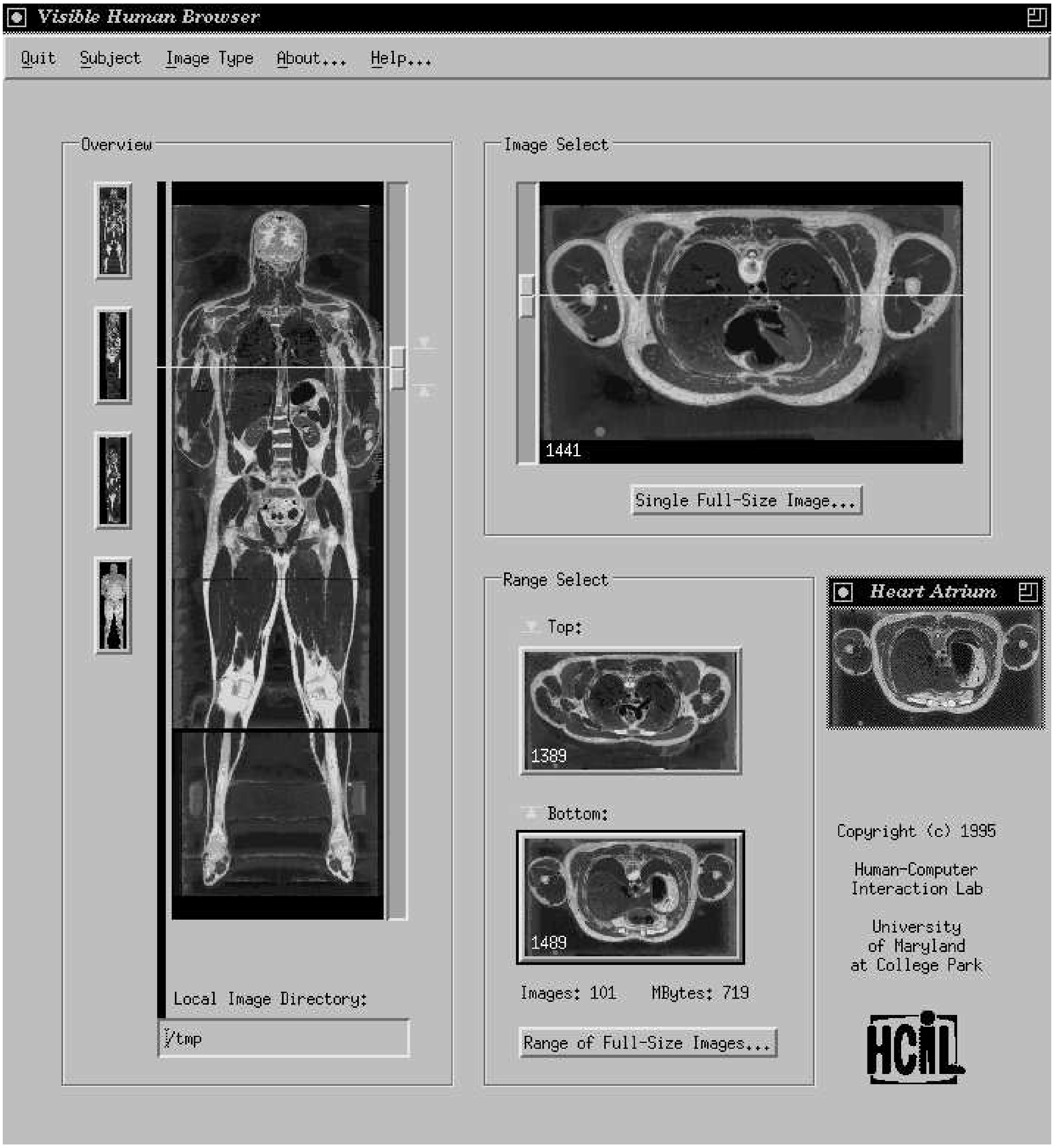

Figure 5: A browser to the Visible Human dataset ready to retrieve cryosections of the heart atrium.

ing old ones, and then saving the result. To do this, run *matrixedit*. It will display the MeSH tree as a node-link tree diagram just as *dotty* does. The first node that is clicked on with the left mouse button will start a new interrelationship group. This is indicated in the yellow color that the node is highlighted. Subsequent mouse clicks will color nodes red and set them to be triggered when the first node is clicked on. A new interrelationship group can be created by holding down the right mouse button and selecting "new group" from the menu that appears. This will restores the nodes highlighted from previous group to the default color.

After all interrelationships have been created, hold down the right mouse button and select "save matrix" from the menu that appears. A dialog box will then pop up for inputting the name of the data file. This is the file that is loaded under "load matrix" in order to view the interrelationships.

Software Architecture

MeSHBROWSE is an application written as a script in *lefty*. *lefty* is an interactive language which, when piped together for interprocess communication with *dot* to draw the graph, provides a way to write callbacks for the WYSIWYG manipulation of graphs. The combination of *dot* for laying out and displaying graphs and *lefty* for responding to mouse and keyboard events forms the system known as *dotty*. Both *dot* and *lefty* were designed and developed by Eleftherios Koutsofios and Stephen North of AT&T Bell Labs and are available under a licensing agreement at http://www.research.att.com/orgs/ssr/book/reuse/.

In order to display the MeSH tree in *dotty* it must be in *dot* format in which the attributes (name, color, size, etc.) are specified for each node and edge, and then the edges are defined as an ordered pair of nodes. However, MeSH in its downloaded format comes as a nested and indented table of contents. The first author implemented a tool in c language called *parsetoc* to convert the TOC to *dot* format, which takes MeSH or some portion of MeSH as input and generates its *dot* equivalent.

There were several advantages to writing MeSHBROWSE in *dotty* rather than using a GUI builder such as Galaxy. First and foremost, *lefty* scripts generally can be put together quickly in this high-level language. The developer need not worry about memory allocation and variable declarations. Compilation time is saved because *lefty* code is interpreted. Another advantage is that, because *dotty* is available as shareware for a variety of platforms, its scripts are easily portable. These conditions are ideal for implementing prototypes.

dotty has several drawbacks, though. Most importantly, it is interpreted, which will make it run too slowly on very large trees. Since *lefty* does not allow for the creation of pointers, the user is very limited in the data structures that can be used within *lefty* code. Also, it does not allow for the creation of widgets such as scrollbars like a UI builder would. The only mouse and keyboard events that trigger callbacks are those on nodes and edges. Therefore, *dotty* is quite insufficient beyond building prototypes.

Query Semantics

The nodes displayed by MeSHBROWSE fall into two categories: entities and attributes. This distinction is somewhat semantic, though. In the congenital heart diseases example, the nodes labeled with names of congenital heart defects can be thought of as the entities while the nodes labeled with structures from the cardiovascular system and the nodes labeled with tissue types can be thought of as the attributes.

Clicking on an entity will trigger its attributes to be highlighted during a chain reaction. In the congenital heart diseases example, the user can click on a specific defect to get some sort of description of it from its attributes. For instance, clicking on "Ebstein's Anomaly" will highlight attributes "Myocardium", "Heart Septum", and "Tricuspid Valve" because they are the structures of the affected pathological region known as cor atrium. A Boolean query formed from a sequence of clicks on entities is useful for discovering overlaps between attribute sets of different entities, perhaps in this case revealing which common structures are affected by two different congenital heart defects.

Clicking on an attribute will trigger the set of entities which have that associated attribute to become highlighted. For instance, clicking on "Tricuspid Valve" will cause "Tricuspid Atresia" and "Ebstein's Anomaly" to be highlighted because they are both congenital heart defects which affect the tricuspid valve. A Boolean query formed from a sequence of clicks on attributes is useful for discovering overlaps between entity sets of different attributes, perhaps in this case revealing similar affects of two different congenital heart defects. This might be helpful for making a diagnosis given a set of symptoms.

Algorithms

When a node is clicked, it is possible for any set of nodes in the tree to be triggered. Thus, interrelationships don't partition the tree into equivalence classes. Clicking on members of triggered set need not by any means trigger the same set of nodes. Formally, if T is the set of nodes in the tree, then

$$f : T \to 2^T$$

where f is the function mapping the clicked on node to the set of nodes in the tree triggered to be highlighted. Note that f need not be one-to-one or onto. There is one constraint to this mapping: $\forall u, v \in T$, if $u \in f(v)$ then $v \in f(u)$. Intuitively, this means that there is a symmetry between any two given nodes (if A triggers B then B triggers A).

We use an adjacency matrix for representing interrelationships as edges in a symmetric directed graph. Since there can be at most one edge in each direction between nodes, a bit matrix can be used to save space in storing the graph. If $|T| = n$ the matrix will be $n \times n$.

In some examples, only a small portion of 2^T is triggered. In many cases

$$f : E \to 2^A$$

where E is the set of nodes considered entities and A is the set of nodes considered attributes. As mentioned in the previous section, this is a somewhat semantic distinction, but knowing this distinction could help us limit the size of the matrix to $|E| \times |A|$. One structural clue to help distinguish might be to assume that a node cannot trigger any of its descendants or ascendants and that only cousin nodes can be triggered. Entities are usually located in one subtree while attributes are located in a separate subtree. This is certainly true in the congenital heart diseases example: the "Congenital Heart Defects" child of MeSH is the parent of a subtree totally separate from the "Cardiovascular System" child and the "Tissue Types" child. This observation was considered in the design of MeSHBROWSE; however, there are advantages to allowing the more general $n \times n$ matrix. Not only does storing the interrelationships in an $n \times n$ matrix make the algorithms easier to implement, but it also allows for nice features like the ability to click on a node to get all of its descendants and their triggers to be highlighted. For example, clicking on "Vena Cavae" would trigger "Vena Cava Inferior" and "Vena Cava Superior", which would in turn trigger the nodes related to "Vena Cava Inferior" plus the nodes related to "Vena Cava Superior".

The following is pseudocode for triggering a chain reaction after a node is clicked for the first click in a query:

```
row ← ID of node clicked on
for each column of bitmatrix[row] do
   if bitmatrix[row, column] = 1 then
      highlight node with ID column
```

For subsequent clicks (performs Boolean AND query):

```
row ← ID of node clicked on
for each node of those currently highighted do
   column ← ID of node
   if matrix[row, column] = 0 then
      restore node to the defaultcolor
```

For subsequent clicks (performs Boolean OR query):

```
row ← ID of node clicked on
for each column of bitmatrix[row] do
   if matrix[row, column] = 1 then
      if node with ID column not highlighted then
         highlight node with ID column
```

The time complexity of each of the above algorithms is $O(|T|)$. Note that these algorithms parallelize well to $O(1)$ time complexity due to the independence of highlighting nodes.

Future Work

The current MeSHBROWSE system is very limited for performing Boolean queries. Only Boolean operations AND and OR have been implemented, and as yet no clever interface for changing between modes has been implemented. MeSHBROWSE could easily be extended for Boolean NOT operations, but, similarly, there is no nice interaction protocol for differentiating between AND, OR, and NOT. It is important to point out that the current implementation does not allow nesting (parenthesizing) of the AND operator and so the only queries that can be performed are of the form $p_1 \wedge p_2 \wedge \ldots \wedge p_n$. Again, there does not seem to be a nice interaction protocol for 'saving' in memory the results of a query in order to AND with a parenthesized part.

Even with these limitations, the appeal of dynamically browsing the hierarchy is great. User engagement with the process of selecting and expanding nodes elicits positive responses and encourages exploration. A next step would be to work with more scenarios and develop a full working system that accessed the image data.

References

[CS94] R. Chimera and B. Shneiderman. An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents. *ACM Transactions on Information Systems*, 12(4):383–406, October 1994.

[Dil90] A. Dillon. Navigation in Hypertext: A Critical Review of the Concept. In *Proceedings of IFIP Interact '90*, North-Holland, 1990.

[Gou68] S. E. Gould. *PATHOLOGY of the HEART and BLOOD VESSELS*. Charles C. Thomas, Springfield, Illinois, 1968.

[KN93] E. Koutsofious and S. North. *Drawing graphs with* dot. AT&T Bell Laboratories, Murray Hill, NJ, October 1993.

[Kou94] E. Koutsofious. *Editing Pictures with* lefty. AT&T Bell Laboratories, Murray Hill, NJ, July 1994.

[Mac84] A. Mackintosh. *The Heart Disease Reference Book*. Harper & Row, London, 1984.

[Mar87] Gary Marchionini. An invitation to browse: Designing full-text systems for novice users. *The Canadian Journal of Information Science*, 12(3):69–79, 1987.

[Mar92] Gary Marchionini. Interfaces for end-user information seeking. *Journal of the American Society for Information Science*, 43(2):156–163, 1992.

[Nat95] National Library of Medicine, Bethesda, MD. *MeSH – Tree Structures*, 1995.

[NK95] C. North and F. Korn. Browsing anatomical image databases: A case study of the visible human. Technical report, *1995 HCIL Video Report. University of Maryland, Department of Computer Science, College Park, MD, 1995.*

[NSP95] C. North, B. Shneiderman, and C. Plaisant. User controlled overviews to explore a medical image digital library: A case study of the visible human. Technical report, University of

Maryland, Department of Computer Science, College Park, MD, 1995.