Visual Web Surfing with Hy$^+$

Masum Z. Hasan, Alberto O. Mendelzon and Dimitra Vista *

Computer Systems Research Institute
University of Toronto
Toronto, M5S 1A1, Canada

Abstract

In this paper we describe how we are using the Hy$^+$ visualization system to visualize the portion of the World Wide Web explored during a browsing session. As the user browses, the Web browser communicates the URL and title of each document fetched as well as all the anchors contained in the document. Hy$^+$ displays graphically the history of the navigation and multiple views of the structure of that portion of the Web. Queries of the visual query language GraphLog can be issued at any time to query and reformat the visualizations.

1 Introduction

Traditional Internet services and resources, such as email, telnet, ftp, gopher, and wais, have been used by the Internet community for years. It was only recently that a new Internet service, the World Wide Web came into existence on the Internet. Browsers for the Web provide a simple user interface to all Internet services and Web resources. These resources are hypermedia documents written in HTML (HyperText Markup Language) and provided by Web servers (httpd). The easy-to-use Web browsers have made both the Web and the Internet popular to every kind of users. The Web is by far the largest and fastest growing information resource available today. Millions of computer systems are part of the network that comprises the Web, and millions of people are using it every day in many countries worldwide. There are also millions of Web *documents* that a user can potentially browse. In this paper we focus on Web browsing tools and the Web of documents browsed by a user during a single browsing session.

The Web browsers allow users to surf through the Web in hypertext style by clicking on the *anchors* of HTML documents. However, as the session progresses, the user may quickly get out of context. During a session, it may be difficult to know exactly where in the Web the user is and how he or she got there [6]. It is difficult to detect if a place has been visited more than once or even to quickly return to an arbitrary place visited before. In the hope of addressing this problem, we have modified and interfaced one of the Web browsers, Mosaic 2.0, with the Hy$^+$ visualization system. Hy$^+$ supports the graphical presentation and visual querying of structured data [1, 2, 4]. In this context, it displays several overviews of the visited portion of the Web as well as a graphical view of the navigation. The need for such overviews of Web browsing sessions is discussed in [7].

The rest of the paper is organized as follows. In Section 2 we discuss the interface of Hy$^+$ with a modified version of Mosaic 2.0. Different Hy$^+$ views of the browsed portion of the Web are described in Section 3. Section 4 discusses the Hy$^+$ queries that create these views as well as other queries that can be used to filter out interesting subgraphs of the browsed Web. In Section 5 we discuss work in progress to extend the system to other aspects of Web visualization. In Section 6 we discuss related work and we conclude in Section 7.

*IBM Contact: Jacob Slonim, Centre for Advanced Studies, IBM Canada.

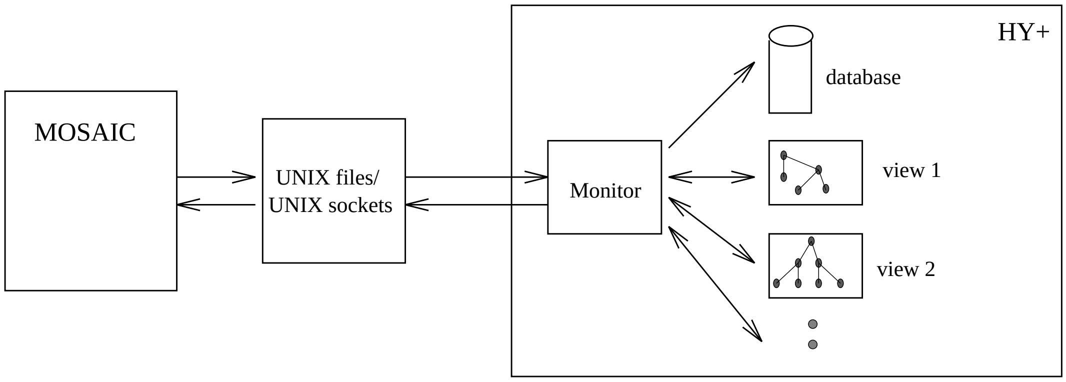

Figure 1: Mosaic interaction with Hy$^+$

2 Interface of Hy$^+$ with a Web Browser

The Hy$^+$ visualization system supports the visual presentation of structured data in the form of *hygraphs* [1, 2, 4]. It provides a number of browsers and graphical editors to display and create the graphical presentations and extensively uses color and icon symbols to improve the generated visualizations. In addition, it has a layout engine that implements several different layout algorithms. Most important, Hy$^+$ supports a visual query language, called GraphLog [3, 5], which allows the specification of queries graphically. GraphLog queries serve a two-fold purpose. They allow the systematic filtering of the data to manage the visual complexity of large datasets, and they allow the specification of new relationships among the data, in much the same way database views do.

Documents accessible from the Web have unique identifiers (addresses) and are essentially hypertext documents containing a number of hypertext links. The user can click on any of these links to go to another hypertext document. Internally, each such link is a pair consisting of the label that can be clicked upon and the address of the document associated with the link. Each such pair is called an *anchor*. An example document address, which is usually called URL (Universal Resource Locator), is

`http : //twist.db.toronto.edu : 8020/Hy.html`

The first part, `http` (HyperText Transfer Protocol), is the protocol used between Web servers and browsers to transfer documents and other information. Existing Web browsers also support other Internet protocols such as `ftp` (File Transfer Protocol), `telnet`, `gopher`, `wais`, and so on. The second part of the address, `twist.db.toronto.edu`, is the name of the machine hosting the Web server, and `8020` is the port number this server "listens" to (the port number is optional). The last part, `Hy.html`, is the document itself. The suffix `html` indicates that the document is written in HTML, the HyperText Markup Language.

We have interfaced the Hy$^+$ visualization system with the Mosaic 2.0 Web browser. As the Mosaic session proceeds, the URL and title of each document fetched as well as all the anchors contained in the document are communicated to Hy$^+$. Hy$^+$, in turn, uses this information to update the different graphical views that it maintains about the sub-web seen so far. These views are internally specified through GraphLog queries and there is no limit as to how many such views a user can have. A database of all the information is maintained so that other queries can also be issued. In addition, all Hy$^+$ visualizations are active; by clicking on a node in a hygraph, the user can make the browser *jump to* the URL associated with the node. Figure 1 shows the interaction of Mosaic with Hy$^+$.

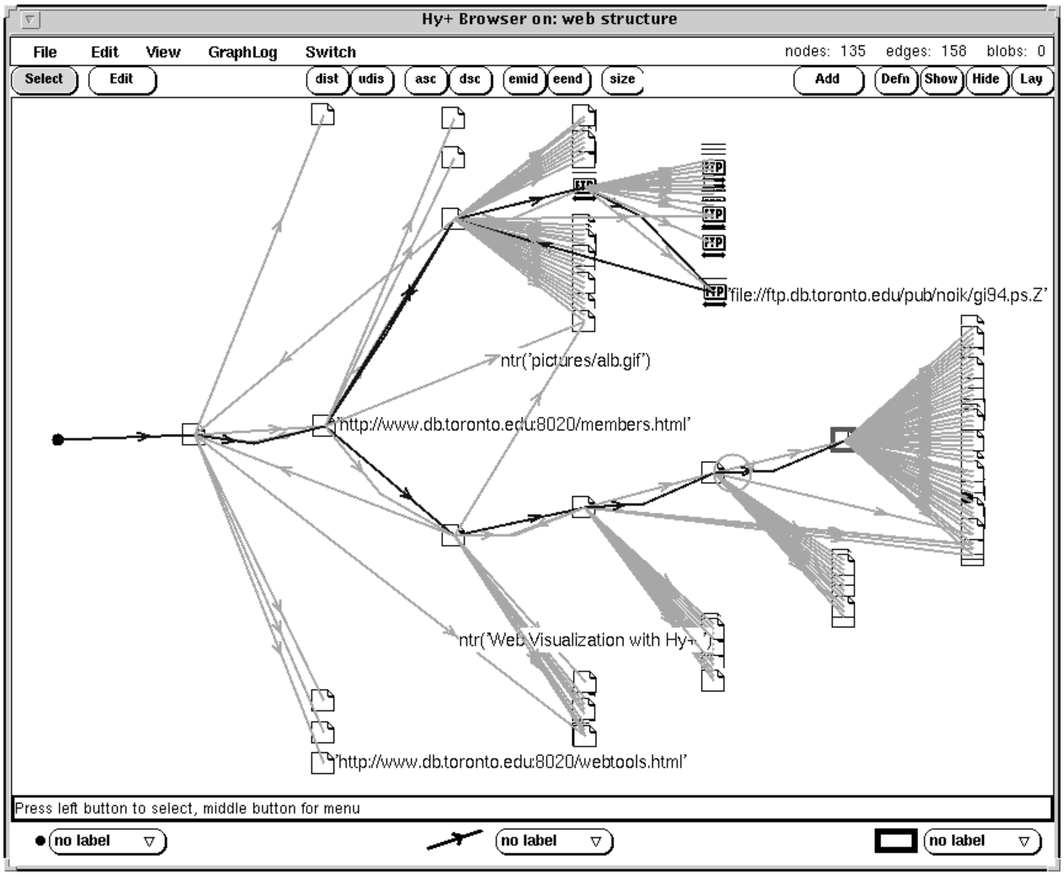

Figure 2: Web structure of a browsing session

3 Hy$^+$ Views

Figure 2 shows a Hy$^+$ generated graphical view of the sub-web seen by the user during a browsing session with Mosaic. In this view, there are two kind of edges, `ntr` (light color) and `tr` (dark color). An `ntr` edge is created from a visited document at URL `a` to any document with URL `b`, if `b` is cited in an anchor within document `a`; the label of the anchor is shown as the edge label, along with the type of edge (`ntr`). A `tr` edge is created from URL `a` to `b`, if `b` is visited immediately after `a`, either by clicking on an anchor inside `a`, or by clicking "back" or "forward", or by opening a new URL location while examining `a`.

The GraphLog query language can be used to restrict the set of documents displayed in this view, using very flexible criteria, including structural properties and regular expression matching on URLs, anchor labels, and document titles. Another advantage of this view is that the user has easy access to a document whose URL is referenced in an anchor within some visited document and can click on it directly, without having to fetch again the whole document that contains the anchor.

The history of visited nodes is shown in Figure 3. This view corresponds (to some extent) to the Mosaic history list. However, Mosaic and other Web browsers do not keep any information corresponding to the `ntr` edges. They only maintain linear history lists that a user can refer to in order to return to a previously visited document. Moreover, these linear histories only provide a path from the origin of the navigation to the current document; all other branches are pruned. Our history graph is more

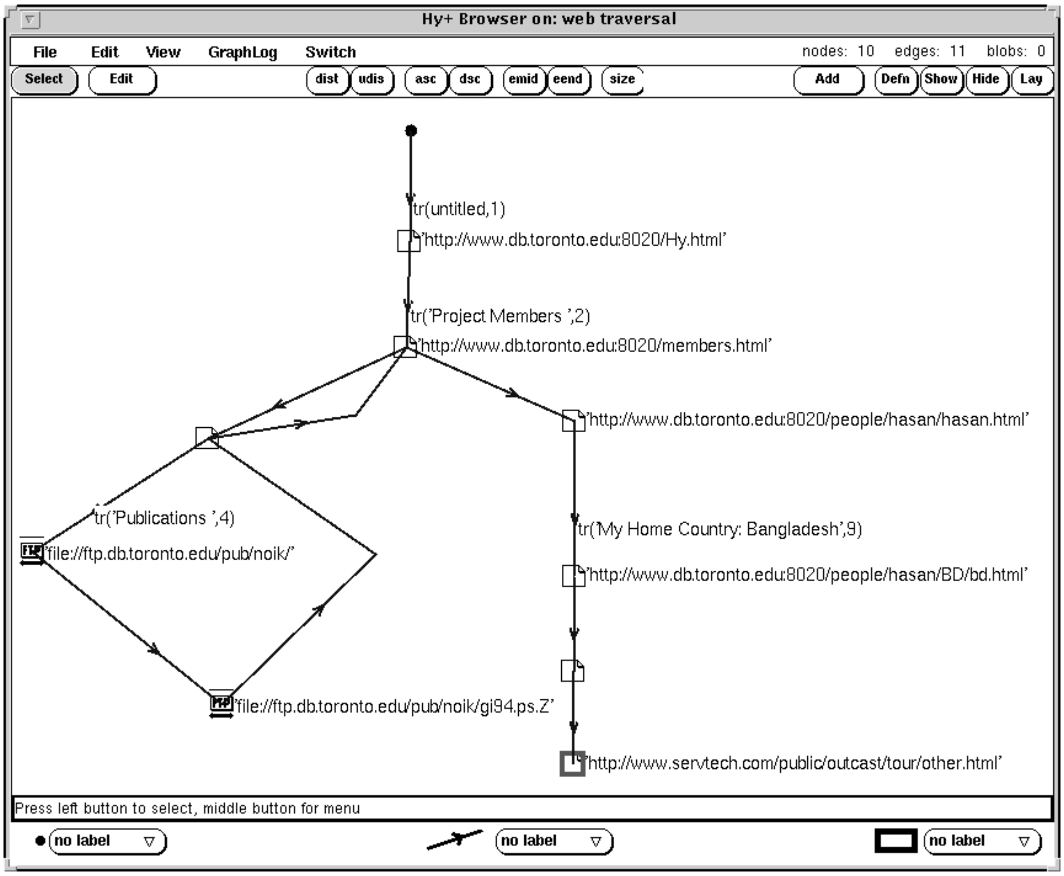

Figure 3: History of visited documents

general and shows more structure than just a linear list of visited nodes. This contextual information can be useful in identifying previously visited nodes that the user wants to revisit. In particular, since the browsing can be controlled from the visualization, clicking on any node causes Mosaic to jump to the associated document.

Figure 4 shows another view of the same Web browsing session as above. This uses the *hygraph* model supported by Hy$^+$, where in addition to associating a node with another node by an edge, any node can be associated with one or more labeled *sets* of nodes by placing these in boxes (called *blobs* in Hy$^+$ parlance). We use these boxes to represent HTML documents; the node at the top left-hand corner of a box is labeled with the title of the document, and each node inside a box represents an anchor (URL/label pair). An edge labeled `by_clicking` (dark color) is created from the anchor that was clicked to the document fetched as a result. The traversal edges from the previous view are also presented (shown in light color) connecting visited documents except that edges produced by hitting the "Back/Forward" buttons are omitted.

Boxes can be interactively collapsed to reduce cluttering. And different icons can be assigned to nodes based on predefined criteria to distinguish different kinds of nodes. For example, in all views different icons are used for different Internet access methods (http, ftp, gopher, and so on).

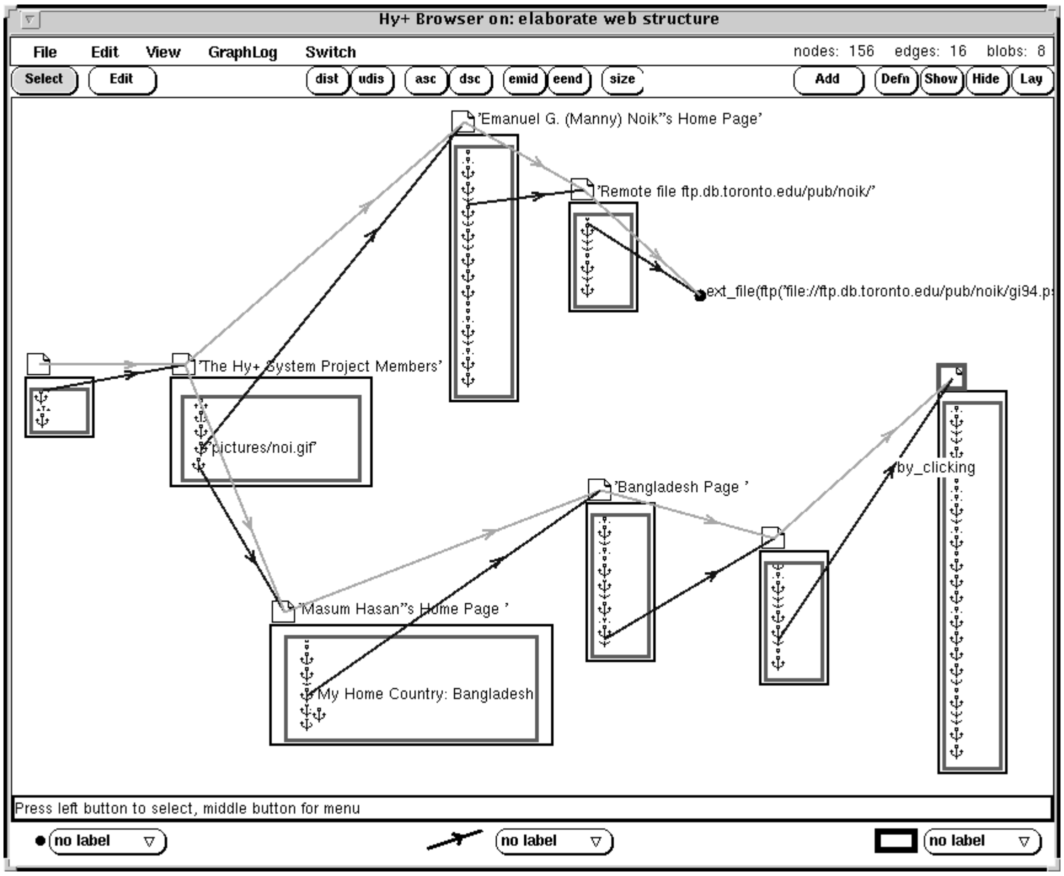

Figure 4: HTML documents viewed as blobs

4 Hy$^+$ queries

The information that is communicated by Mosaic to Hy$^+$ has the form of prolog facts. Every time a document is fetched, its URL, title, and all contained anchors are communicated to Hy$^+$. This information is read by a monitor within Hy$^+$ which dispatches the work that needs to be done. First, the read information is appended into a database maintaining the entire history of the above information. Then, all GraphLog queries generating the graphical views of these data are evaluated to produce the views. And finally, these views are automatically updated on the screen to reflect changes.

The following is the schema of the relations that Mosaic has been modified to provide:

- `tr(URL1, URL2, Label, Step)` indicates that a traversal from `URL1` to `URL2` was done in Mosaic by clicking the `Label` in `URL1`'s text (HTML text/image). This traversal is numbered `Step`, a number that is used in the visualization of the traversal to distinguish the order in which the traversal occurs;
- `ntr(URL1, URL2, Label)` indicates that within the document addressed by `URL1` there is an anchor with label `Label` pointing to `URL2`;
- `trav(doc(URL1,Title1), doc(URL2, Title2))` indicates that a traversal occurred from the document whose address is `URL1` and whose title is `Title1` to another document with address `URL2` and title `Title2`;

- `contains(doc(URL1, Title), anchor(URL2, Label))` indicates that in the document with address `URL1` and title `Title` there is an anchor associated with `Label` pointing to the document with address `URL2`;

- `by_clicking(anchor(URL, Label), doc(URL, Title))` indicates that either the `Label` has been clicked upon from the Mosaic screen while a document containing the anchor `anchor(URL, Label)` was being viewed or that the node corresponding to this anchor has been clicked upon from a Hy$^+$ visualization, causing a traversal to the corresponding document with address `URL` and title `Title`.

As mentioned above, each of the generated views is produced by evaluating a GraphLog query. In GraphLog there are two kinds of queries: *filter* and *define*. Filter queries request a subset of the database facts to be retrieved. Such queries are included in a box called `showGraphLog`. Define queries create new relations from the existing data. Define queries are included in a box called `defineGraphLog`.

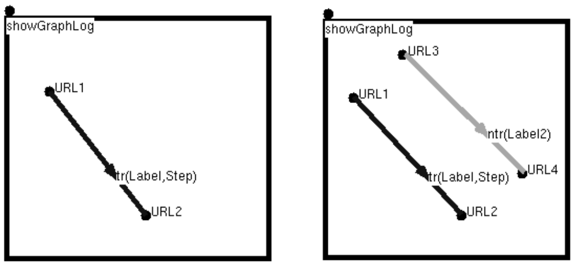

Figure 5: Queries generating the views of Web traversal and Web structure

The two queries that produce the first and second view of Mosaic data are shown in Figure 5. The first is a filter query that retrieves all known `tr` facts from the database (the traversal information). Note that, although `tr` was introduced as having four arguments (two URLs, one label and one incremental number), when `tr` is displayed as an edge (in the query as well as in all visualizations), it has only two arguments (the label and the number). This is the convention in Hy$^+$: when the Prolog fact $p(X_1, X_2, X_3, X_4)$ is visualized as a graph edge, the edge goes from a node labeled X_1 to a node labeled X_2 and it is labeled $p(X_3, X_4)$. The second query in the figure is a filter query that retrieves both the `tr` and `ntr` edges from the database (the traversal together with the containment information).

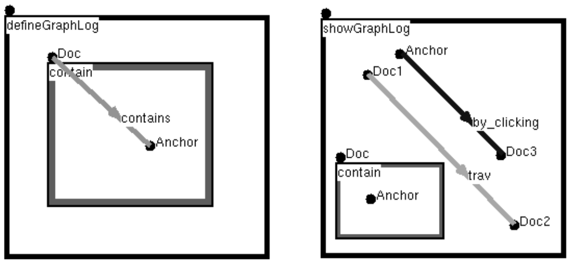

Figure 6: Query generating the blob view of the Web structure

The query that produces the third view is shown in Figure 6. It has two sub-queries. The define sub-query changes the visualization of the containment edges into blobs. For every `contains` edge from a node labeled `doc(URL1, Title)` to a node labeled `anchor(URL2, Label)`, all the anchor nodes are collected in a blob associated with the containing document. The blob thus defined is named `contain`. The filter sub-query retrieves these blobs together with the `trav` and `by_clicking` relations.

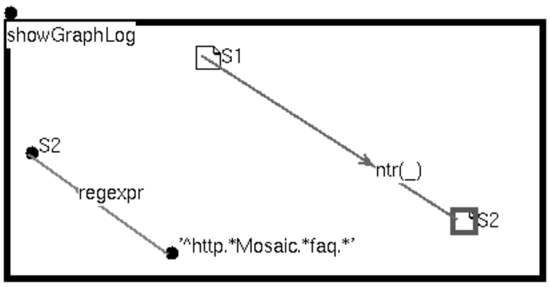

Figure 7: Query requesting sites on frequently asked questions about Mosaic

Further queries can be executed on the collected data. Figure 7 shows such a query. It is a filter query that asks for all URLs that start with the word `http` and contain the words `Mosaic` and `faq`. That is all

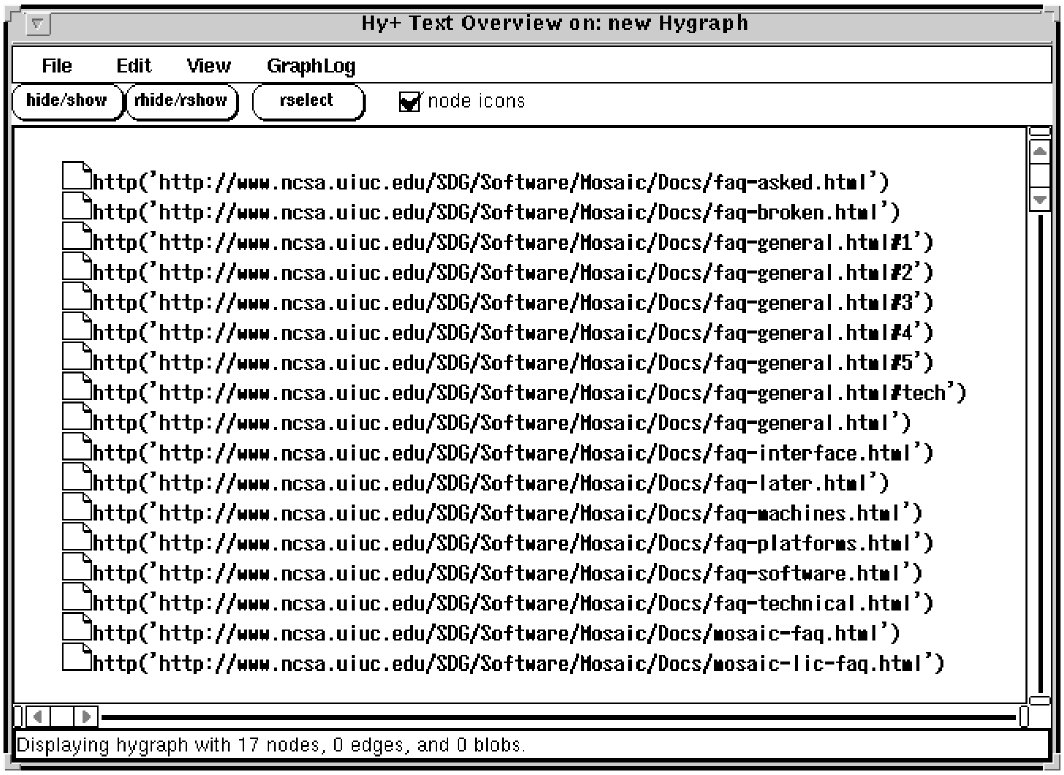

Figure 8: Sites on frequently asked questions about Mosaic

http documents, `S2`, such that `ntr(S1, S2)`, and `S2` is described by the regular expression `ĥttp.*Mosaic.*faq.*`. The intention of the query is to filter ULRs collected during the navigation for all the places that contain "frequently asked questions" (`faq`) about the Mosaic browser. Note that, in this query the edges are not thick as before but a node, `S2`, is distinguished. This is because we are not interested in the edges themselves but rather in the nodes these edges lead to. Also note that, the `ntr` edge in this query does not specify any arguments in it, but rather uses an underscore, which in Prolog-like languages is used to denote an anonymous variable. Evaluating this query on some data that was produced during another Mosaic session produces the answer shown in Figure 8.

5 Extensions to the System

In this section we discuss our ongoing work on extensions to the above visualizations and on other aspects of Web visualization.

5.1 Graphical View of Document Structure

Consider the visualization shown in Figure 4. Inside each box (blob) we have visualized only the anchors contained in the corresponding document. However, we can also visualize the hierarchical structure of the document. Clicking on a node in this hierarchy allows the portion of the document corresponding to the node to be shown on the Web browser. For example, consider the following HTML document (modified for the paper).

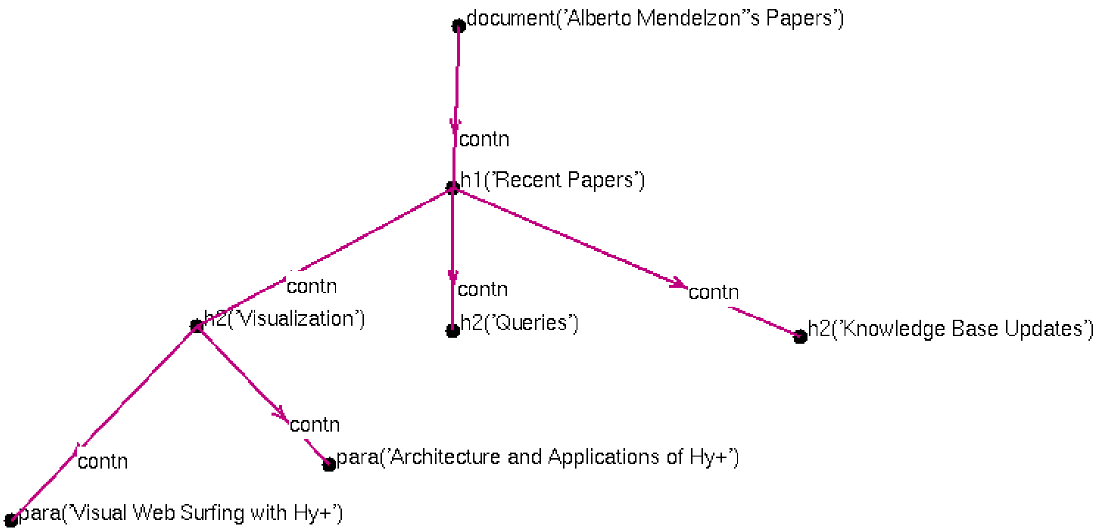

Figure 9: Document structure visualization

```
<html>
<head>
<TITLE>
Alberto Mendelzon's Papers
</TITLE>
</head>
<body>
<hr>

<h1>
Recent Papers
</h1>
<hr>

<h2> <a name="visualization">
     Visualization </a> </h2>

<p>
<a href="p1.Z">
Visual Web Surfing with Hy+
</p>

<p>
<a href="p2.Z">
Architecture and Applications of Hy+
</p>
:
:

<h2> <a name="queries">
     Queries </a> </h2>

:
:

<h2> <a name="updates">
     Knowledge Base Updates </a> </h2>

:
:

<hr>
</html>
```

This document has been converted automatically into the graph shown in Figure 9. Informative labels gleaned from the HTML document are assigned to each node. Clicking on **h2(''Visualization'')** will show the entire subtree rooted at that node in the browser's page as an HTML document. Clicking on a **para** node, indicating a paragraph, will show only that paragraph. To advance this idea further we are also working on a graph-based authoring tool that will show graphs of documents as in Figure 9 and allow users to cut and paste subgraphs from existing documents.

5.2 Query-directed Mapping of Web Space

All visualizations discussed above deal only with a single browsing session. A user may be interested in mapping a portion of the Web and visualizing the resulting map. We distinguish two kind of maps: *server map* and *link map*. A server map consists of a Web server and all the documents residing in the server. A link map is a map of documents accessible by following links from a certain set of documents.

We are working on a *Web query language* which will allow users to specify flexible queries to build such maps.

6 Related Work

Very few tools have been developed to address the problem of visualizing the Web.

The system that to some extent deals with the same problem to the one discussed here is the KJ-Editor developed by a collaboration of the Toyohashi University of Technology and Keio University in Japan [7]. With the KJ-Editor, the user can construct an overview diagram of the Web. This diagram is constructed by the user, who for each page visited in the Web browser, decides whether to add it to the overview [7].

Webspace, which is being developed at the Geometry Center of the University of Minnesota, is a tool that visualizes the hypertext links between HTML documents. However, it does not visualize the sub-web seen by the user during a browsing session. It starts with a URL and constructs a map of hypertext links up to a certain depth. It uses an interactive tool (popularly termed *robot*) to fetch Web documents. Clicking on a node in the view fetches the associated document, as in our views.

WebViz [8] is a system for server access log analysis through visualization, developed at the Graphics Visualization & Usability Center at the Georgia Institute of Technology. It provides the user with graphical information about document accesses and the possible paths taken by users.

7 Conclusion

We have described a system that allows a user of a Web browsing tool to get graphical overviews of his or her browsing session. The system provides views of not only the history of the navigation, but also the containment of links in visited documents, even if these are not followed in the navigation. These views are generated by executing queries of the graphical query language GraphLog. We have also shown how GraphLog queries can be used to further query the generated visualizations. All the views in the system are active, that is, a document can be fetched on the Web browsing tool by clicking on the corresponding node in some view. Our system is unique in that it is based on a general purpose database visualization system instead of ad hoc techniques.

About the Authors

Masum Z. Hasan is a Research Associate at the Computer Systems Research Institute at the University of Toronto and a PhD candidate in the Department of Computer Science at the University of Waterloo. He obtained BEng, MEng in Computer Engineering from the former USSR and MMath in Computer Science from the University of Waterloo. His research interests are in active temporal databases, network management, networked document browsing/searching, distributed and parallel programming environment, and visualization.

Alberto O. Mendelzon is a Professor in the Department of Computer Science at the University of Toronto. His interests include the theory, practice and applications of databases and knowledge bases, and in particular the data visualization and data manipulation interfaces for them.

Dimitra Vista is a PhD candidate in the Department of Computer Science at the University of Toronto. Her interests are in the area of query processing, query optimization, incremental query evaluation, and Web visualization.

References

[1] M.P. Consens. *Creating and Filtering Structural Data Visualizations using Hygraph Patterns.* PhD thesis, Department of Computer Science, University of Toronto, February 1994.

[2] M.P. Consens, F.Ch. Eigler, M.Z. Hasan, A.O. Mendelzon, M.G. Noik, A.G. Ryman, and D. Vista. Architecture and Applications of the Hy$^+$ Visualization System. *IBM Systems Journal*, 33(3):458–476, 1994.

[3] M.P. Consens and A.O. Mendelzon. GraphLog: A Visual Formalism for Real Life Recursion. In *Proceedings of 9th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 404–416, 1990.

[4] M.P. Consens and A.O. Mendelzon. The Hy$^+$/GraphLog Visual Query System. In *Proceedings of the ACM-SIGMOD 1990 Annual Conference on Management of Data*, page 388, 1990. Video presentation summary.

[5] M.P. Consens, A.O. Mendelzon, and D. Vista. Deductive Database Support for Data Visualization. In *Proceedings of the 4th International Conference on Extending Database Technology*, pages 45–58, 1994.

[6] J. Nielsen. The Art of Navigating Through Hypertext. *Communications of the ACM*, 33(3):296–310, March 1990.

[7] J. Nielsen. *Multimedia and Hypertext, The Internet and Beyond.* AP Professional, first edition, 1995.

[8] J.E. Pitkow and Bharat K.A. WebViz: A Tool for World Wide Web Access Log Analysis. In *Proceedings of the First International World-Wide Web Conference*, Geneva Switzerland, May 1994.