A Simple but Effective Web-Based Report Server

Liu Xiong* with Michael J. Dinneen†
Department of Computer Science
University of Auckland
Private Bag 92019
Auckland, New Zealand

Abstract

We present a practical web-based server that supports several views (entries sorted by author, title and serial number) of a collection of research reports. A powerful search feature, which exploits Perl regular expressions, is incorporated in the server. Although designed for the Centre for Discrete Mathematics and Theoretical Computer Science (Auckland, New Zealand), this simple presentation tool can be easily installed to disseminate electronic copies of technical reports for other institutions. By design, no HTML programming experience is necessary for adding new reports to the server (or browser interface).

1 Introduction

Every year, the Centre for Discrete Mathematics and Theoretical Computer Science (CDMTCS) adds a few research reports to its collection. The previous dissemination system collects these reports in a single static HTML web page. As the number of reports increases, the plain system needed to be modified to handle a large number of reports. In order to manage these reports more efficiently and elegantly, a web CGI-based research report presentation system was developed to upgrade the old system. The new system interacts between the Web user (with a browser) and web server using CGI Perl scripts to create the dynamic HTML to display reports according to requests from users.

This paper introduces the new report server. In the next section we give an user's guide to the basic functionality of the system. We then in Section 3 explain how to do searching with the use of Perl regular expressions. Finally, in Section 4 we describe the web server aspects of the report presentation section.

*`lxiong@xtra.co.nz`
†`mjd@cs.auckland.ac.nz`

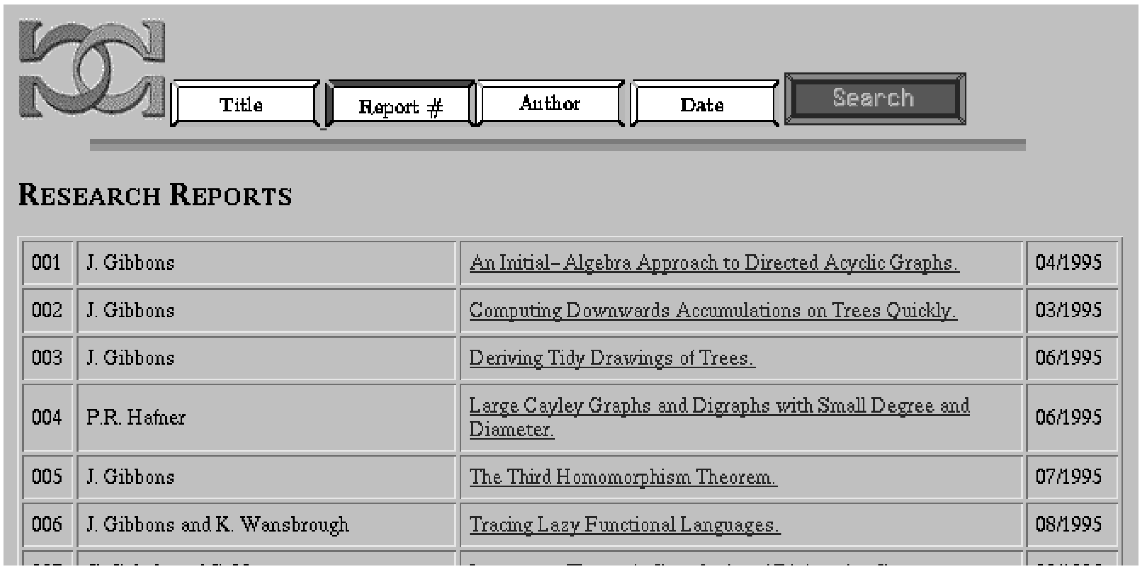


Figure 1: The first page or the page when the serial button is pressed.

2 An User's Guide

The research report presentation system can be used to refer the specified reports in a very efficient way. It supplies convenient and powerful sorting, grouping and searching functionalities.

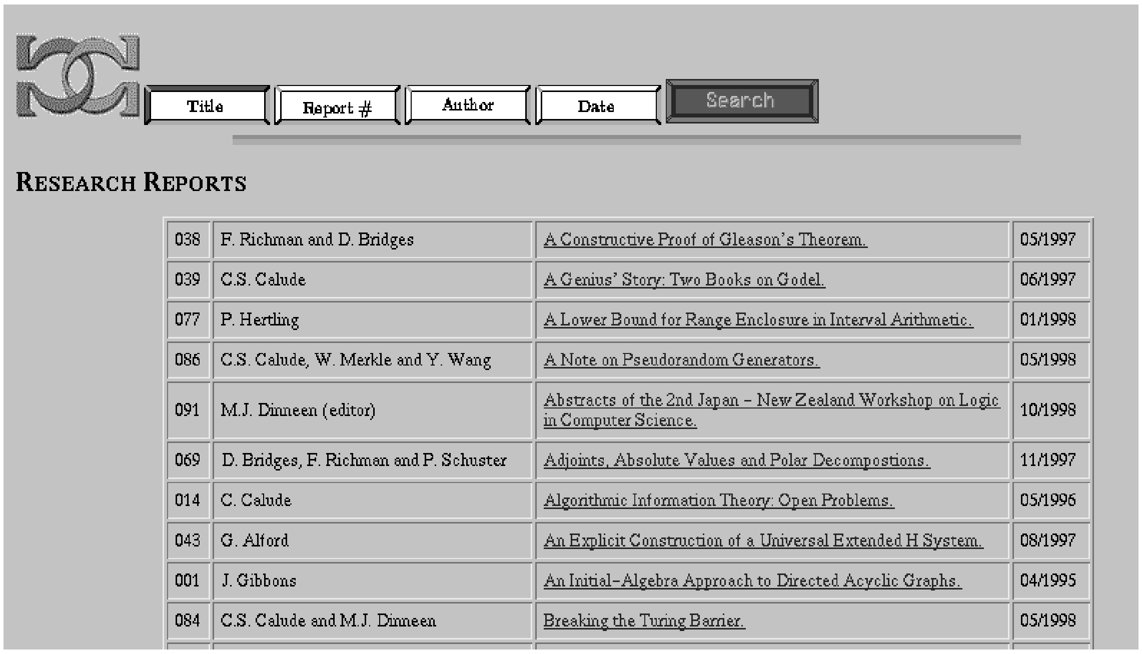
When enter this system, the first page would be displayed as follow: As we can see, there are five action buttons in the title bar: title, author, serial, date and search. A list of all the reports is displayed in a table. By default setting, reports are sorted by their serial numbers.

Screen button specification:

- **Basic sorting**. First, this system implements the basic sorting mechanism. A "serial" and a "title" buttons are supplied to handle this option.
  - "title" button.

    This button provides the functionality of sorting reports by their titles. After the "title" button is clicked, we can see that all reports are listed. Of course they are sorted by their titles. When this button is set, it becomes disabled. A result of this action is shown in Figure 2.
  - "serial" button.

    Each reports is assigned a unique serial number. After the "serial" button is clicked, a new page would come out with all the reports sorted by their serial numbers. This button is also set as default, shown as in figure 1. The "serial" button is disabled if it is selected.
- **Group reports by author or date**. The second functionality of this system is that it allows users to group reports by each author and the year when they



Title | Report # | Author | Date | Search

RESEARCH REPORTS

038	F. Richman and D. Bridges	A Constructive Proof of Gleason's Theorem.	05/1997
039	C.S. Calude	A Genius' Story: Two Books on Godel.	06/1997
077	P. Hertling	A Lower Bound for Range Enclosure in Interval Arithmetic.	01/1998
086	C.S. Calude, W. Merkle and Y. Wang	A Note on Pseudorandom Generators.	05/1998
091	M.J. Dinneen (editor)	Abstracts of the 2nd Japan – New Zealand Workshop on Logic in Computer Science.	10/1998
069	D. Bridges, F. Richman and P. Schuster	Adjoints, Absolute Values and Polar Decompostions.	11/1997
014	C. Calude	Algorithmic Information Theory: Open Problems.	05/1996
043	G. Alford	An Explicit Construction of a Universal Extended H System.	08/1997
001	J. Gibbons	An Initial-Algebra Approach to Directed Acyclic Graphs.	04/1995
084	C.S. Calude and M.J. Dinneen	Breaking the Turing Barrier.	05/1998

Figure 2: The page displayed after the "title" button is clicked.

are issued. an "author" button and a "date" button are supplied to handle this operation.

- "author" button.

  The "author" button is applied to group reports by their authors. When this button is clicked, a sorted list of names of all the authors would be displayed as shown in figure 3. There is a small icon in front of each author name. If we keep going to click on this icon, a table would be pop out which contains all the reports of this selected author. If the icon is clicked again, the table would be closed. Figure 4 and Figure 5 demonstrate this process.

  Same as other buttons, the "author" button is disabled after it is clicked.

- "date" button.

  The "date" button is used to group reports by the year when they are issued. The "date" button has the same operations as "author" button.

• **Searching**. This system provides powerful and efficient searching among different fields. We will discuss this mechanism in next section.

• **Output table format**. Each report can be output in a HTML table. It occupies one row in this table. Each table entry contains four fields: Serial number, Author, Title, and Date, where the title field is linked to the location for the original paper. Therefore by clicking the title field, we can access and refer to the original report.

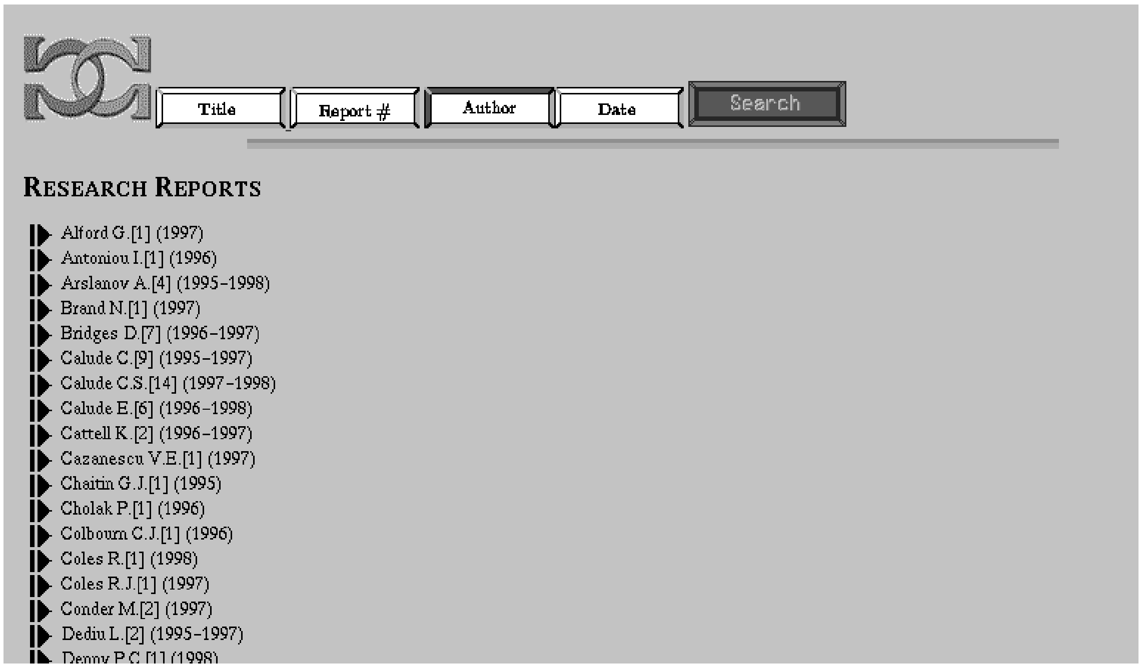


Figure 3: The page for grouping reports by author.

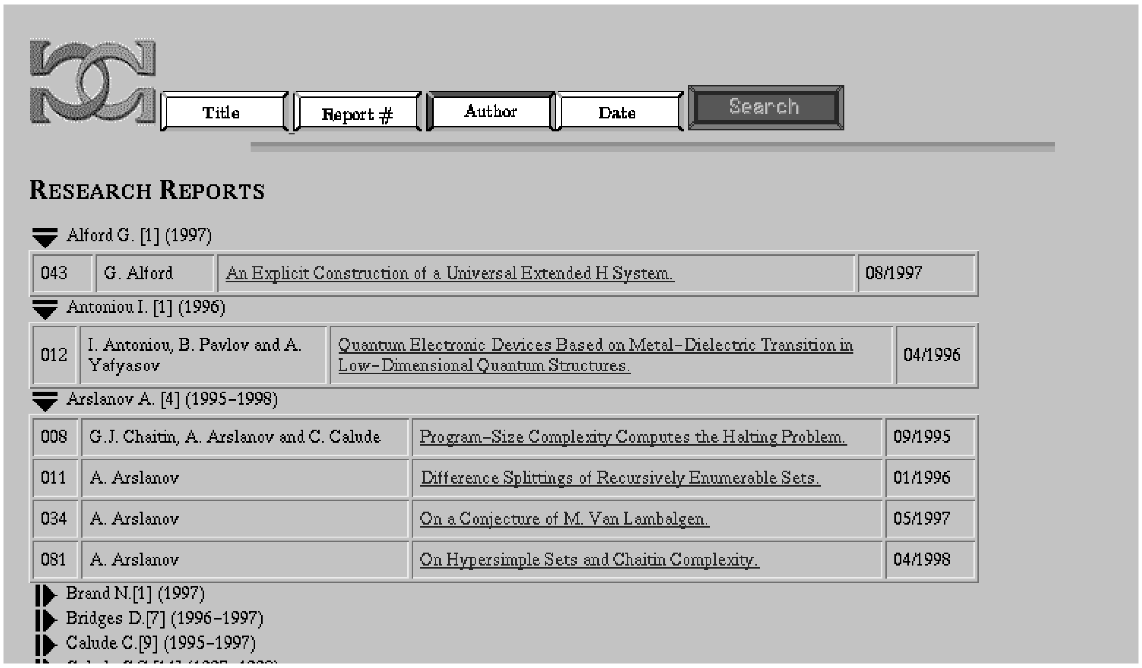


Figure 4: The page for grouping reports by authors with some authors selected.

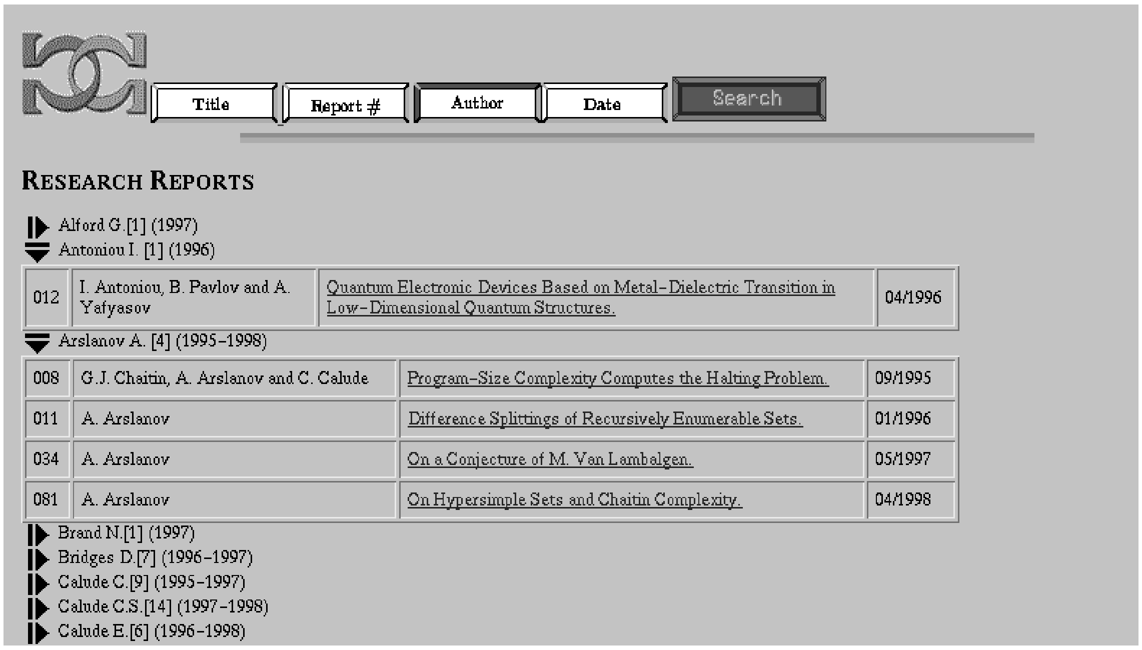


Figure 5: The page for grouping reports by authors with one author closed again.

3 Searching using Perl's Regular Expressions

This presentation implements general searching mechanism and allows regular expressions as the searching pattern. The second Perl script mysearch.pl is written for this function.

When the "search" button is pressed by users in a page, a new HTML page with a searching form will be displayed as follow:

In this form, a searching operation is allowed in four field: title (search by title), author (search by author), and keyword (search by keyword). This operation may or may not be case-sensitive under the control of a checkbox.

As a general searching operation, a text pattern can be used to match into the database to fetch desired data in this system. The general searching is very popularly implemented at other online searching servers. Here we don't want discuss it too much. Our goal for this section is to introduce the powerful regular expression searching function.

First, a question should be asked: what is a Perl regular expression? Regular Expressions are the most obvious very high-level feature of Perl. A single pattern match in Perl-even a simple one-can perform the work of many lines in a different languages. Pattern matches, especially when combined with Perl's handling of strings and lists, provide capabilities that are very difficult to mimic in other programming languages.

Regular expressions in Perl are made up of atoms. Atoms are connected by operators like repetition, sequence, and alternation. Most regular expression atoms are single-character matches. The rules of the regular expressions matching are specified as follow:

Each character matches itself, unless it is one of the special characters + ? . * ^$ () [] | \. The special meaning of these characters can be escaped using a backslash \ .

. Matches an arbitrary of patterns to a single element match.

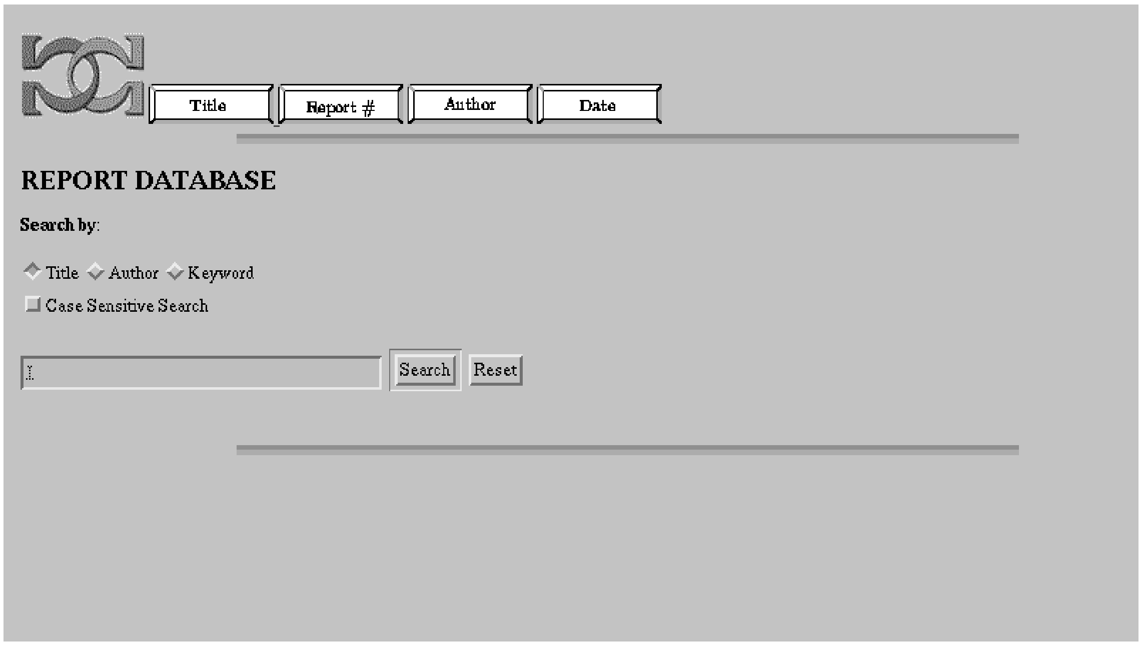


Figure 6: The form for searching reports.

(...) Groups a series of pattern elements to a single element.

^ Matches the beginning of the target. In multi-line mode also matches after every newline character.

$ Matches the end of the line. In multi-line mode also matches before every newline character.

[...] Denotes a class of characters to match. `[^]` negates the class.

(...|...|...) Matches one of the alternatives.

(?:# text) Comment.

(?: regexp) Like (regexp) but does not make back-references.

(?=regexp) Zero width positive look-ahead assertion.

(?! regexp) Zero width negative look-ahead assertion.

(?! modifier) Embedded pattern-match modifier. modifier can be one or more of i, m, s, or x.

+ Matches the preceding pattern element one or more times.

? Matches zero or one times.

* Matches zero or more times.

{n,m} Denotes the minimum n and maximum m match count. {n} means exactly n times; {n,}, means at least n times.

\w Matches alphanumeric, including_, \W matches non-alphanumeric

\s Matches whitespace, \S matches non-whitespace.

\d Matches numeric, \D matches non-numeric.

\A Matches the beginning of the string, \Z matches the end.

\b Matches word boundaries, \B matches non-boundaries.

\1 ... \9 ... Matches the *i*-th saved expression in ()'s.

Here are some examples:

ab*c Matches ab, abc, abbc, abbbc, etc.

abc* Matches ab, abc, abcc, abccc, etc.

ab(c)* Same thing, and memorizes the c actually matched.

abc{?: c} * Same thing, but doesn't memorize the c .

abc{2, 4} Matches abcc, abccc, abcccc.

(abc)* Matches empty string, abc, abcabc, etc.

ed|jo Matches ed or jo.

(ed) | (jo) Same thing.

ed|jo1,3 Matches ed, jo, joo, jooo.

^ed|jo$ matches ed at beginning, jo at end.

^(ed|jo)$ Matches exactly ed or jo.

Hence users can apply the above rules to do searching. For more on how to use the regular expression, a Perl Quick Reference can be referred [5]. The following figures show some searching examples by using regular expressions as patterns.

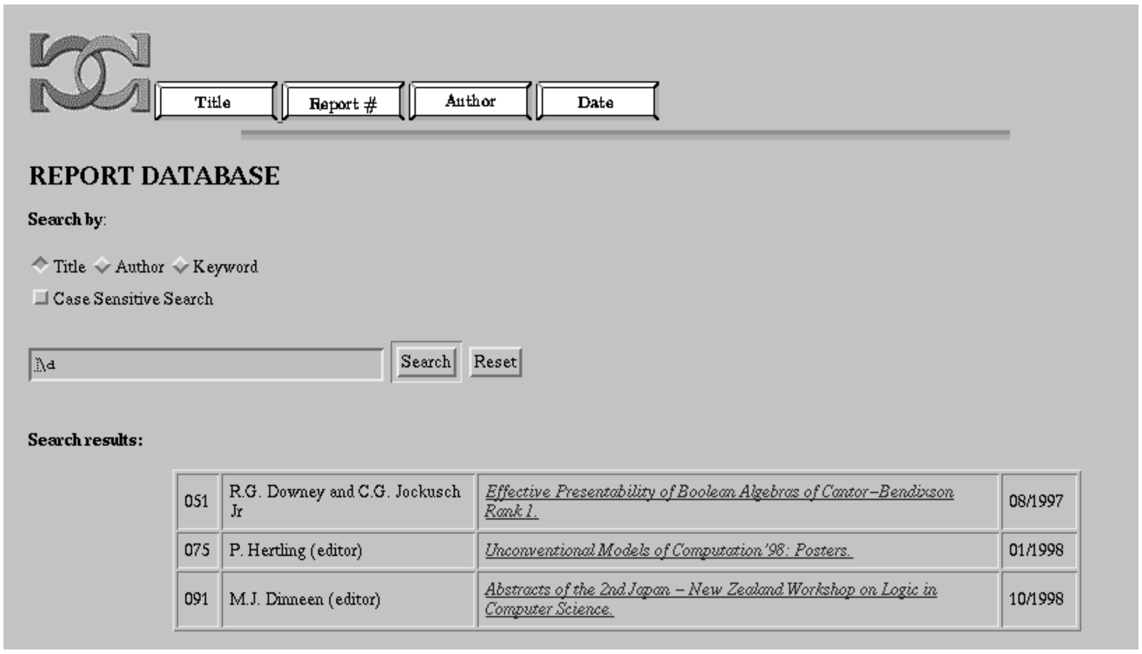


Figure 7: A regular expression to search all the reports have numbers in their tiles.

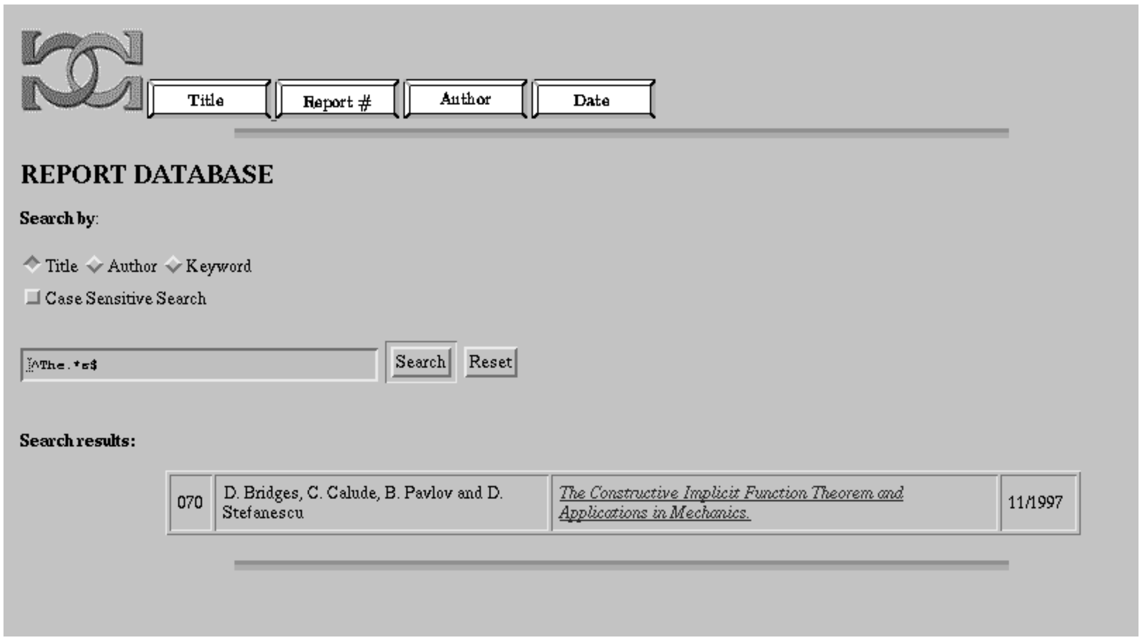


Figure 8: A pattern to search those reports which have a title start with "The" and end with "s".

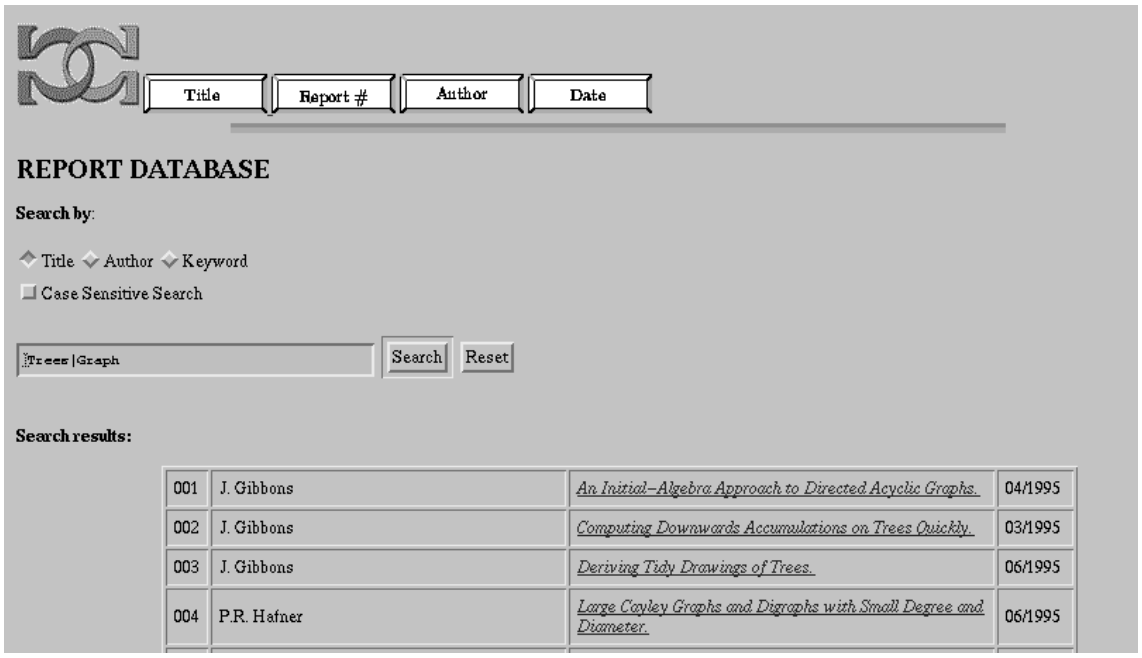


Figure 9: The regular expression "Trees|Graph" is used to search all the report with their title containing either "Trees" or "Graph".

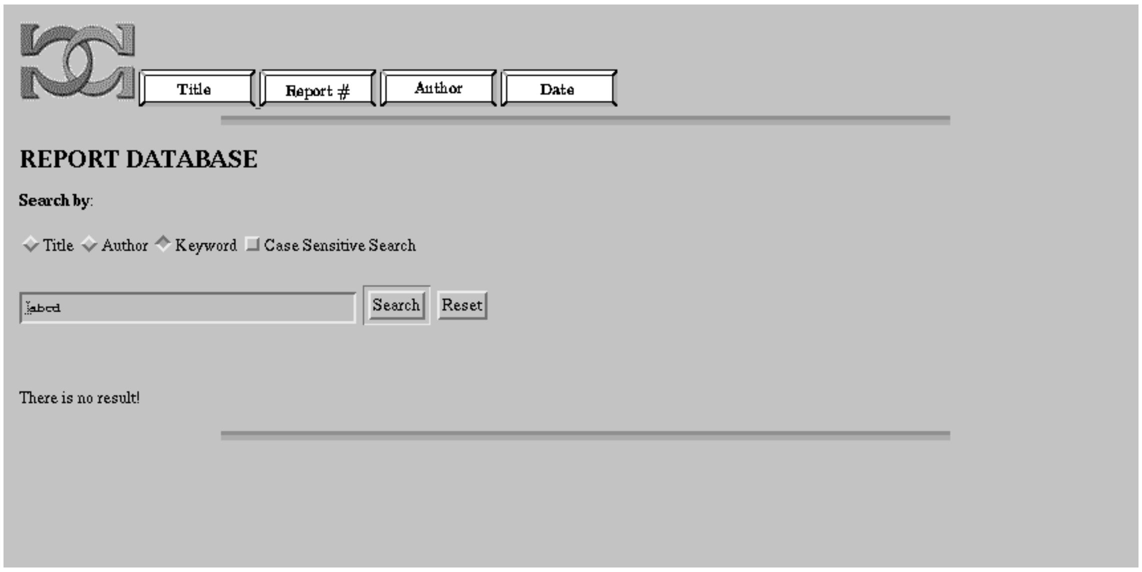


Figure 10: If a pattern can't be matched in any reports, there should be no result.

4 Report Server Schematics

The presentation system includes four sections: HTTP server, Research report database, CGI scripts and Web browser.

- **HTTP Server**

  The Hypertext Transfer Protocol (HTTP) is the language that Web clients and Web servers use it to communicate with each other. It is essentially the backbone of the Web. In this system, the HTTP server responds to call the CGI script, translate user specified operations (defined in CGI script) and pass them as parameters into a running script and fetch information from the database back to the script.

- **Research report database**

  Currently, the research report database is a plain text file storing the information about each reports in CSMTC center. Each report has a serial number, a list of authors, a Web location, a title and a published date as its attributes. They occupy one line in the database and are separated by a hash symbol "#". Why the fields of a report must be separated by a delimitation? This is because CGI Perl scripts have a cool split function which can take the delimitation as the pattern, splits a string into an array of strings, and returns it. So that the user specified data can be retrieved.

  A simple example of this database is showed as follow:

  001#J. Gibbons#001damgs.pdf#An Initial-Algebra Approach to Directed Acyclic Graphs#04/1995

  002#J. Gibbons#002quickly.pdf#Computing Downwards Accumulations on Trees Quickly#03/1995

  003#J. Gibbons#003drawing.pdf#Deriving Tidy Drawings of Trees#06/1995

  004#P.R. Hafner#004cayley.pdf#Large Cayley Graphs and Digraphs with Small Degree and Diameter#06/1995

  To update the database, we just simply input the new reports information following the same format. So that this system can be maintained without any program and HTML skills.

  The format of the database is very essential. Especially, there can not be any empty line in the end of the file. Otherwise, some unexpected symbols would be output.

  As you can see, a plain text file is simple to implement, however it has capacity and performance limitations when dealing with large amount of information. Hence a SQL database can be a useful choice. There are a few commercial SQL database packages available on the market. Under Linux, for example you have MySQL a shareware product, and it is very powerful too.

  In order to port existing design into a SQL database environment, CGI scripts will have to communicate with the chosen SQL database. This communication can be achieved through ODBC (Open Database Connectivity) layer.

Within a SQL database environment, multiple tables can be set up to store information more effectively. For example, an author table can have a lot more information about authors, such as an unique author ID and contact details. A publication table needs to have author's ID on each record in order to join these two tables together during search. The search result can then provide all necessary information about a particular author if required.

- **Web Browser**

  The Internet Web Browser is the underlying interface between users and the system. It not only receives user specified data and pass them to the system through HTTP server, but also returns the results to users by displaying on the screen. In our system, Netscape or Internet Explorer can be set as the browser.

- **CGI Perl scripts**

  The kernel part of the system is two CGI Perl scripts. All the functionalities are handled by them. To understand these scripts, let's review some CGI concepts.

  - **What is CGI?**

    The Common Gateway Interface(CGI) is an essential tool for creating and managing comprehensive Web sites. With CGI, you can write scripts that create interactive, user-driven applications.

    CGI is the part of the Web server (in our system, that is HTTP) that can communicate with other programs that are running on the server. With CGI, the Web server can invoke an external program, while passing user-specific data to the program (such as what host the user is connecting from, or input the user has supplied through an HTML from). The program than processes that data and the server passes the program's response back to the Web browser.

  - **Why Perl?**

    Why do we use Perl to write the CGI scripts? Practical Extraction and Report Language (Perl) is an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It's also a good language for many system management tasks. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal).

    Perl has three basic data types: scalars, arrays of scalars, and hashes of scalars, also known as associative arrays. Scalars are the fundamental type from which more complicated structures are built. A scalar stores a single, simple value, typically a string or a number. Elements of this simple type can be combined into either of the two composite types. An array is an ordered list of scalars that you access with a numeric subscript. A hash is an unordered set of key/value pairs that you access using strings(keys) as subscripts, to look up the scalar value corresponding to a given key. Variables are always one of these three types.

In our scripts, Array and Hash data are used to achieve retrieving information from texture Database with high performance. By the way, the charming Perl functions, split and sort, play the important roles to handle these data structures. The split function splits a string into an array of strings, and returns it. The sort function sorts a list (array or hash) and returns the sorted array value. However, a cool Perl 5 library CGI Lite are also used to decode both URL-encoded and multi-part form data produced by the file upload feature present in the Web browser.

On the other side, we also need to mention the unique and mysterious Perl regular expression operations, especially the Pattern Matching mechanism. With the help of some magic Perl regular expression operators such as "=~", "\s","\t", we can match any text pattern into the database and grab the desired data. As we can see, our system has the major features as other online servers, but the Perl scripts only have several hundreds of lines. These all come from the contribution of the Perl language.

– **Scripts specification**

  * **secondcgi.pl**
    This script handles the sorting and grouping functionalities. The algorithm for this script is as follow:

    · The default commands are specified by the CGI cookies (see next section) which store in the browser.

    · The user specified commands are input as a QUERY_ STRING and passed to this script by HTTP server.

    · The script opens the database and stores it into an array.

    · If the command is "serial", the script transfers reports into an associative array hashed by their serial number and applies Perl function "sort" to sort the hash keys(serial number) and then outputs the sorted reports to a HTML file.

    · If the command is "title", same as d, the titles are specified as the hash keys rather than the serial numbers.

    · If the command is "author", the script stores reports into an associative array hashed by the names of authors and sorts the hash keys (sorted by authors), outputs the sorted keys(a list of authors) into a HTML file. If an author is further clicked, then takes the author name as a hash index, fetches all the reports of this author from the hash array and outputs a new HTML file with all the reports issued by this author listed in a table which following the name of the author. If an "open" author name is clicked, then the list of reports is "closed" in another HTML page.

    · If the command is "date", applies the same statements as "author" with the hash key instead by a year number.

    · If the command is "search", calls another script "mysearch.pl" to do the searching.

Note: in this script, some server functions, such as “setText” and “setCookie” are included to interact with the HTTP server.

* **myserach.pl**

  This script implements searching mechanism. Especially, supplies more powerful and efficient searching approaches to users by allowing fancy Perl regular expressions.

  The algorithm for this script is as follow:

  - Outputs a form which let the user input a specified searching key and a searched text.
  - Decodes user specified data which contains the searching key and the searched text by using CGI Lite library functions: “parse_form_data”. The “parse_form_data” method parses the form data and stores it in an internal associative array which can be placed in a variable defined by a Perl script.
  - Uses the “`=~`” operator to match the searched text into the database by the specified searching key, outputs the reports which contains the searched text into browser.

- **How to keep system persistent state?**

  The other important feature of the presentation system is maintaining state. The CGI cookies are employed to implement this feature. Client side cookies are introduced to enable a server to store client-specific information on the client's machine, and use that information when a server of a particular page is accessed again by the client. The cookie mechanism allows servers to personalize pages for each client, or remember selections the client has made when browsing through various pages of a site - all without having to use a complicated (or more time-consuming) CGI/database system on the server's side.

  Cookies work in the following way: When a CGI program identifies a new user, it adds an extra header to its response containing an identifier for that user and other information that the server may glean from the client's input. The header informs the cookie-enabled browser to add this information to the client's cookies file. After this, all requests to that URL from the browser will include the cookie information as an extra header in the request. The CGI program uses this information to return a document tailored to that specific client. The cookies are stored on the client user's hard drive, so the information remains even when the browser is closed and reopened.

5 Conclusion

By the above discussion, the research report presentation system has the following advantages:

- Simple and can be used conveniently. The system provides a very flexible interface to each user.

- Very easily to be maintained. Because all HTML pages are created dynamically and automatically, and the database update dependents only on simple text editing, the system can be maintained without any HTML and programming skills.
- Powerful. This system covers major features provided by other online servers. It features an attractive searching operation by allowing regular expressions.

Therefore this system is a very useful tool for navigating a repository of web-based research reports.

References

[1] S. Gundavaram. *CGI Programming on the World Wide*, O'Reilly & Associates, Inc., 1996.

[2] J. N. Hall and R. L. Schwartz. *Effective Perl Programming: Writing better programs with Perl*, Addison-Wesley, 1998.

[3] A. Homer, C. Ullman and S. Wright. *Instant HTML Programmer's Reference*, 2nd edition, Wrox Press, 1997.

[4] S. Spainbour and V. Quercia. *Webmaster in a Nutshell.* O'Reilly & Associates, Inc., 1996.

[5] L. Wall and R. L. Schwartz. *Programming Perl*, O'Reilly & Associates, Inc., 1991.

[6] O. Kirch. *Linux network administrator's guide*, O'Reilly & Associates, Inc., 1995.

[7] P. Palmer. *The Web server handbook*, Upper Saddle River, N.J.: Prentice Hall PTR, 1996.

A Report Server Code

On request, the CGI script for the web server is available by e-mail from the second author of this paper.