ABSTRACT

A TOOL FOR PHYLOGENETIC DATA CLEANING AND SEARCHING

by
Viswanath Neelavalli

Data collection and cleaning is a very important part of an elaborate Data Mining System. 'TreeBASE' is a relational database of phylogenetic information at the Harvard University with a keyword based searching interface. 'TreeSearch' is a Structure based search engine implemented at NJIT that can be used for searching phylogenetic data. Phylogenetic trees are extracted from the flat-file database at Harvard University, available at {ftp://herbaria.harvard.edu/pub/piel/Data/files/}. There is huge amount of information present in the files about the trees and the data matrices from which the trees are generated. The search tool implemented at NJIT is interested in using the string representation of the trees for query and retrieval of information.

The purpose of this thesis and the work related to it, is to make an automated tool to clean the files present in the Harvard University's Database using pattern-matching techniques and gather all the phylogenetic trees' string representation to build a local database of clean phylogenetic data that can be readily used in a versatile fashion.

A TOOL FOR PHYLOGENETIC DATA CLEANING AND SEARCHING

by
Viswanath Neelavalli

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

January 2002

APPROVAL PAGE

A TOOL FOR PHYLOGENETIC DATA CLEANING AND SEARCHING

Viswanath Neelavalli

Dr. Jason Tsong Li Wang, Thesis Advisor — Date
Professor of Department of Computer Science
College of Computing Sciences, NJIT

Dr. Chengjun Liu, Committee Member — Date
Assistant Professor of Department of Computer Science
College of Computing Sciences, NJIT

Dr. Qicheng Ma, Committee Member — Date
Novartis Pharmaceutical Corporation

BIOGRAPHICAL SKETCH

Author: Viswanath Neelavalli

Degree: Master of Science

Date: January 2003

Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2002

- Bachelor of Science in Mechanical Engineering specializing in
  Production Engineering, Chaitanya Bharathi Institute of Technology, Osmania University, Hyderabad, India, 2000

Major: Computer Science

Presentations and Publications:

G. Chandra Mohan Reddy, Viswanath Neelavalli, et al.
"Computer Aided Experimental Analysis of Drawing a Cup using a Blank Holder", All India seminar on "Condition Based Monitoring", The Institution of Engineers (India), Orissa State Center, June 1999.

Viswanath Neelavalli and Vinodh Putarjunan
"Expert Systems and Artificial Intelligence",
University of Roorkee, UP, India, February 1999.

Viswanath Neelavalli and Vinodh Putarjunan
"Applications of Expert Systems in Manufacturing Industry",
Birla Institute of Technology, Pilani, India, February 1999.

Viswanath Neelavalli and Vinodh Putarjunan
"Rapid Prototyping",
Chaitanya Bharathi Institute of Technology, Hyderabad, India, November 1999.

To my beloved family

ACKNOWLEDGEMENT

First, I would like to thank my thesis advisor, Dr. Jason Tsong Li Wang, for his invaluable direction, assistance, and guidance. In particular, his timely recommendations and suggestions have been very crucial for the rapid progress and improvement of the work. I also thank my committee members for their continuous support although my research.

I also thank Mr. Muralidhar Maddala, my erstwhile Java Instructor, for the numerous invaluable techniques of programming he had taught me and also for his continuing guidance. I also thank my parents and my dear brother, in India, for their immense support all through my work. Thanks are also due to Ms. Katherine G Herbert, my laboratory's System Administrator, and my laboratory colleagues for their invaluable support throughout the work. Special thanks to my friends, Geeta Josyula, Kalyan Kuchimanchi and Vijay Mareddy, who helped me in many ways.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the rapid technological advancement in the field of generation and collection of data over the past few decades, enormously large data banks have been built. The use and retrieval of such stored data generates the need for new techniques and automated tools that can intelligently assist us in transforming the vast amount of data into useful information and knowledge. This leads to the concept of Data Mining also popularly known as *Knowledge Discovery in Databases (KDD),* which has seen some rapid progress in the past few years.

There is a greater need to develop much efficient data preprocessing techniques than ever before. Real-world databases are highly susceptible to noisy, missing and inconsistent data due to typical huge size, often in gigabytes or more. Data needs to be preprocessed in order to help improve the quality of the processing and consequently, of the *mining* results.

The are a number of data preprocessing techniques, some of which are outlined as follows-

Data Cleaning: to discard noise and remedy inconsistencies in the data

Data Integration: to merge data from multiple sources into a coherent data store like a data warehouse or data cube

Data Transformation: like normalization, used to improve the accuracy and efficiency of the mining algorithms involving distance measurements

Data Reduction: used to reduce the data size by aggregating, eliminating redundant features or clustering for example. [1]

This work is an effort to clean and filter the noisy data in the database of phylogenetic information, TreeBASE, sponsored by Harvard University Herbaria, Leiden University EEW, and the University of California, Davis. TreeBASE is a relational database designed to manage and explore information on phylogenetic relationships. Its main function is to store published phylogenetic trees and data matrices. It also includes bibliographic information on phylogenetic studies, and details on taxon, characters, algorithms used, and analyses performed [5]. The main interest of this work is to extract the relevant phylogenetic information and prepare a local database that has data

- which can be searched flexibly
- which can be converted into any convenient format for further processing.

CHAPTER 2

CURRENT RESEARCH

Phylogenetics is the branch of study dealing with the evolutionary development and history of a taxonomic grouping of organisms, at large, to the development of any organ or a part of an organism or the development of a tribe or racial group, altogether. There is an amazing diversity of life, in both living and extinct. The research issues in this area vary from classification or categorization of the species into an organized hierarchy by identifying the patterns and processes involved in evolutionary change, to development of a standardized phylogenetic nomenclature keeping in view the ever growing need to match the phylogenetic knowledge with the taxonomy of all the species known thus far. There is a great deal of research in identifying patterns in the existing databases of phylogenetic information to predict the behavior of less known species and also about the species yet to be discovered. Phylogenetic databases, thus offer a great deal of potential for research for developing analysis and search tools of the biological data. This chapter will delve mainly on the prominent phylogenetic analysis tools as well as well-known databases around the globe.

2.1 CAIC

CAIC (Comparative Analysis by Independent Contrasts) is a user-friendly computer tool for the Apple Macintosh PCs that finds and calculates phylogenetically independent contrasts in one or more variables, enabling the user to test hypotheses of correlated evolution. The tool can be found at the Internet URL http://www.bio.ic.ac.uk/evolve/software/caic/index.html [17]. Closely related species

tend to be similar because of shared inheritance, rather than through independent adaptation. Ordinary statistics such as correlation and regression cannot validly be used with comparative data.

The tool is distributed as a binhexed, self-extracting archive and includes two helper programs for creating phylogenies. The tool TreeEdit could be used to translate trees from other formats like NEXUS. The tool was initially written by Andy Purvis in 1991 at the Evolutionary Biology Group, Oxford University. The second version of the work tool was developed by Andrew Rambaut using Think Pascal 4.5 and the latest version 2.6x has been developed by Nick Issac and Paul-Michael using Code-Warrior Pro 4 at the Imperial College. The tool runs on any Macintosh system.

The dataset for the tool can have up to 10,000 taxa and 128 columns of data. Up to 20 columns of data can be selected in any one analysis. Each tip must be separated from the root of the phylogeny by no more than 49 nodes.

2.2 MacClade

MacClade is a tool used for phylogenetic analysis. The tool has great analytical strength in the studies of character evolution. It also provides many tools for entering and editing data and phylogenies, and also for producing tree diagrams and charts.

The tool is maintained by David R. Maddison and Wayne P. Maddison. The tool is available at the internet URL http://phylogeny.arizona.edu/macclade/macclade.html [18]. The tool provides an interactive environment for exploring the phylogeny. The tool also provides a 'Tree Window' where the phylogenetic trees or cladograms could be manipulated and

character evolution visualized. A summary of changes in the characters can also be depicted in the window and as the trees are manipulated, the tool updates the statistics and the results are updated in graphics and charts.

The tool provides charts for depicting the summary of character evolution on more then one tree and also it provides charts comparing two or more trees. The tool also provides a systematic and comparative data-editor with versatile features of graphical manipulation and also for convenient usage of various rows, columns and blocks of data. The tool also facilitates interoperability between other prominent software called PAUP; facilitating management of files stored in NEXUS format and also helps in calculating the decay indices of the trees by storing the files in a format that can be understood by PAUP.

2.3 PHYLIP

It is a package of programs for inferring phylogenies. The package is freely available on the Internet at http://evolution.genetics.washington.edu/phylip.html [19] as a number of executables. Joe Felsentein of the Department of Computer Science at University of Washington maintains the web page. The package can be run on various platforms like the Windows 95/98/NT, DOS and also Macintosh systems.

The programs contained in the package facilitate wide of variety of analysis that include parsimony, distance matrix, and maximum likelihood methods, including bootstrapping and consensus trees, Invariants (or Evolutionary parsimony) analysis, interactive tree manipulation, compatibility analysis, comparative method analysis, and tree plotting or drawing. The collection of programs can operate on a wide variety of

data types that include molecular sequences, gene frequencies, restriction sites, distance matrices, and 0/1 discrete characters.

The package of programs can be conveniently downloaded from the web by ftp. The web site also provides links to a large set of phylogenetic analysis programs and databases over the Internet. The resources are very conveniently categorized on the basis of the various functions they provide like General-Purpose packages, Parsimony programs, Distance Matrix methods, Quartets Methods, Artificial Intelligence methods, Interactive Tree Manipulation methods, to quote few.

2.4 TreeBASE

TreeBASE is one of the fastest growing databases of phylogenetic data. TreeBASE can be accessed on the web at http://www.treebase.org/treebase/index.html [5]. It is a relational database containing phylogenetic information obtained from research papers submitted to the web site. The site also allows the users to search the database. It also allows the user to gain access to information concerning tree as well as use comparison tools to learn more about various taxa contained within the tree and their relationships with other taxa within the database. TreeBASE is currently maintained by Michael Donoghue, William Piel, Mike Sanderson, and Mary Walsh.

TreeBASE mainly consists of the phylogenetic trees submitted to it by the authors of the papers that present the trees. The site accepts, for review, any peer-reviewed and published paper that provides information on any kind of phylogenetic trees. The paper goes through a review process before being incorporated in the database.

The search techniques TreeBASE employs are based on keyword queries. The interface facilitates the search of the database on taxa, author, citation, study accession and matrix accession number and also structure. The search results comprises of other resources for further analysis of the trees. These other resources include the ability to "tree surf", download the matrix which models the tree, and initiate a structural search of the tree by providing a link to TreeSearch, a structural search engine maintained at NJIT by Dr.Jason T.L. Wang and Dr.Dennis Shasha at http://aria.njit.edu/~biotool/search_index.html [20]. The matches for any keyword contain information about the study in which the keyword was found. The information also includes the publishing date, the Author, the title of the study in which the keyword was found, and the periodical in which the study appeared. Analyses of the data presented in the study are also present in the results. These analyses could include the matrix, a link to the pictorial representation of the tree in a frame, a link for downloading the tree so that the user can view it on his or her own viewer, and a link to bookmark the view for quick retrieval.

TreeSearch takes the format of representation of phylogenetic data from TreeBASE. TreeSearch also obtains its database from TreeBASE. TreeBASE has incorporated into the functionalities of its tool a link to TreeSearch, if a user wants to perform a structural search on the database.

2.5 TREEMAP

TREEMAP is a tool that facilitates visual comparison of phylogenetic trees of host and parasite organisms. It is developed and maintained by Roderic D. M. Page at the Division of Environmental and Evolutionary Biology at the Institute of Biomedical and Life Sciences at the University of Glasgow and can be found at http://taxonomy.zoology.gla.ac.uk/rod/treemap.html [21]. The tool is available for both Macintosh and Windows platforms.

The TREEMAP provides a versatile user interface to explore the relationship between the phylogenies of the host and parasite organisms by facilitating various manipulations. The tool primarily consists of four windows for versatile display of information. The first window, called the Tanglegram, displays the host and the parasite trees showing the interdependencies by connecting lines. The appearance of the trees could be altered, for better understanding, by rotating descendants around their ancestors. The trees can be viewed in either cladogram or phylogram formats and can only be viewed in the phylogram format if the data contains information about the branch lengths. The fonts and colors can also be modified according to the viewer's choice.

The second and the most important window, called the Reconstruction window, helps in demonstrating how the nodes of the parasite tree can be assigned to the nodes of the host tree to create the best reconstruction for those trees. The window displays the parasite tree overlaid on the host tree for easy understanding of the reconstruction. The facility of randomization help the user to evaluate the relationships between the host and parasites and the result of the randomization is displayed in the Histogram

window. The fourth window, called the Branch Length window, uses a bivariate plot for plotting the lengths of branches corresponding to the host and parasite trees.

On Macintosh machines, TREEMAP accepts data in TREEMAP format, flat file text format, and PICT format. The *.PICT formats can be edited with any Macintosh graphics or word processor. For Windows, the TREEMAP data files are ASCII files with the extension *.NEX and picture files are generated with the extension *.WMF. The *.WMF files can be edited using a graphics or word processing tools in Windows environment. The tool provides a very user-friendly interface with versatile features for comparative study of phylogenies of host and parasite organisms.

2.6 Ribosomal Database Project II

RDP, a set of programs, is developed by and is maintained by the Ribosomal Database Project at Michigan State University [Rib]. The resource is available on the web at http://rdp.cme.msu.edu/html/index.html [22] from which the tools can either be downloaded or used online. The RDP provides services for online data analysis of ribosomal information.

The RDP-II's tool set of nine programs contains many helpful tools for analyzing ribosomal information. Firstly, the 'Alignment Slices' presents a compressed view of (up to five) vertical alignment slices, with identical neighboring sequences represented only once. The program proves useful in examining probe candidates, covariation or for browsing a region of an alignment. The 'Chimera Detection' program helps in determining if a sequence is composed of two fragments that are similar to

clearly different sequences from the database and uploaded user sequences, i.e. if the sequence is of chimeric origin. The 'Probe Match' helps evaluating a specified probe sequence with rRNA sequences in the RDP database. The analysis consists of an overview of the matches between a probe and its potential target as a list and has a phylogenetic overview. The 'Sequence Align' accepts sequence entries and aligns them against the most similar RDP sequence. This program highlights unambiguous alignment regions, with ambiguous or uncertain alignment regions displayed in non-highlighted text. The 'Similarity Match' measures similarity between the user's sequence(s) and those in RDP. The 'Similarity Matrix' computes a similarity/dissimilarity matrix for the user's sequences and the most similar database sequences in the RDP database. The '(sub) Tree' is an interactive Java applet that can be used as an *output device* to display results of a query, or as an *input device* to obtain a selection of organisms from the user. The program has been optimized to handle large trees, but it is not an editor that would let the user change tree topologies. The T-RFLP program facilitates comparative community analysis using terminal restriction fragment length polymorphisms. It runs as a java applet from the resident web page.

2.7 SYSTERS

It is a protein database with a versatile tool for evaluating phylogenetic information. This tool uses a web interface to evaluate nucleic and protein information using set of tools that the user can select to use. It is maintained by Antje Krause and can be found on the World Wide Web at http://systers.molgen.mpg.de/ [23].

The SYSTERS Cluster Set is based on the SYSTERS (SYSTEmatic Re-Searching) algorithm. The algorithm finds a set of related sequences (called Cluster) that share a strong similarity to the original query sequence. The sequences are not ranked further within the cluster.

The tool provides the user with a wide variety of tools to search and analyze the clusters. Some of the tools include selecting the clusters by Cluster number, Accession number, also available a re t ools S SMAL a nd B LAST w ith us er inp utted p rotein o r nucleotide sequences. The tool also facilitates the use of other tools and formats of data including PROSITE numbers, PDB Accession numbers, ENZYME numbers, and also accession numbers and identifiers of databases like PIR, SWISS-PROT, FlyBase. The tool also supports multiple formats of query strings.

The tool proves to be a powerful means for evaluating protein and DNA sequences through cluster evaluation because of the provisions in the tool for working with various formats of queries and also providing links to various related databases. Allowing a multitude of formats at many different stages of the search facilitates understanding of the clustering information.

2.8 The Tree Of Life Project

'Tree of Life' is a collection of about 2000 World Wide Web pages, authored by biologists around the globe, providing information about the diversity of life. Pages are linked in to each other in the form of evolutionary tree of organisms with links leading to the pages about the group's subgroups. The resource is accessible on the World Wide

Web at http://phylogeny.arizona.edu/tree/phylogeny.html [24]. The project is currently being maintained by David R.Maddison at University of Arizona.

The Tree of Life Project comprises of numerous websites that are authored through a contribution process. Usually, a researcher who coordinates a certain group within the project also authors the basal pages of the group. The coordinator selects authors for his / her particular set of websites. Anyone interested in coordinating or donating illustrations is encouraged to contact the project.

The website apart from providing links to the facilities it offers, it also provides the user with a huge resource of links relating to the research round the globe in the same field. One of the main resources on the main page is the link to 'browse' web page that provides links to the key pages of the project. It contain the links to the 'rootpage', that provides links to all the other pages, sample pages that demonstrate the diversity of a select few organisms in the project, 'popular groups' providing a simple hierarchy of the popular groups of organisms on the project, 'picture sampler' that provides convenient display of pictures of the required clade or picture that are randomly chosen and finally a link to the search tool of the project.

The search tool of the project is based on a keyword search technique. The tool can search the database for taxon names, clade names, or any of the author names, or references to retrieve a set of links to all the pages that contain the specified keyword.

The project is a great repository of exhaustive information relating to wide variety of life forms. It serves as a great reference for research and also teaching. The project does not facilitate comparative study of life forms in its current stage. In future,

it is intended to implement the project in dynamically generated web pages that could facilitate the comparative study.

2.9 Summary

The research in this area is ever widening as we keep increasing our knowledge about the origin and evolution of life on Earth. There are many analysis tools and databases being used for phylogenetic research to be described in this small chapter. The current effort is to contribute in the direction of having better tools of phylogenetic analysis.

CHAPTER 3

PHYLOGENETIC DATA

3.1 Overview of Phylogenetic Data

Phylogenetic data can be data relating to evolutionary development of some organ or part of an organism or that of a species altogether. The data is mainly in the form of trees and data matrices that are published regularly in leading journals [4]. It is also represented in the format of trees. An example of the string representation of phylogenetic data referred earlier in the document, is shown below:

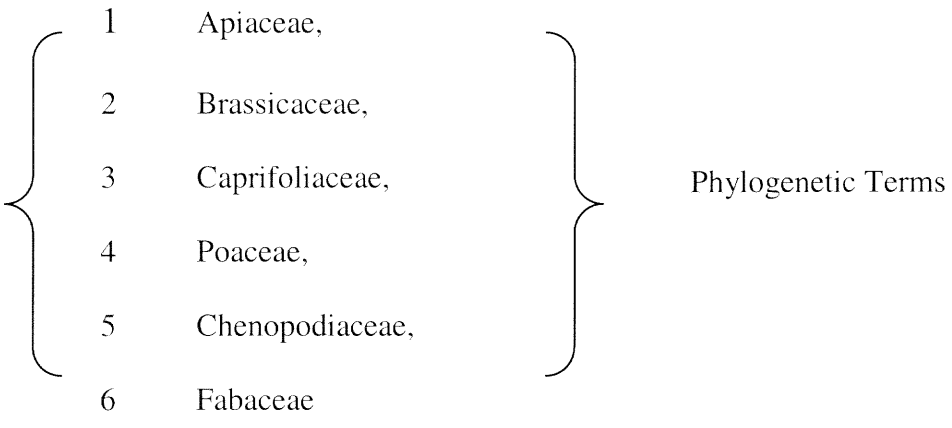

Fig. 3.1 Phylogenetic Terms

```
TREE Fig._6 =  [&R] (4,((1,5),((2,6),3)));

TREE Fig._7 =  [&R] (4,(((1,5),3),(2,6)));

TREE Fig._8A =  [&R] (((1,5),(2,6),3),4);

TREE Fig._8B =  [&R] ((1,2,3,5,6),4);
```

Fig. 3.2 Phylogenetic Tree Representation

The string representation is as shown below:

```
TREE Fig._6 = [&R]
(Poaceae,((Apiaceae,Chenopodiaceae),((Brassicaceae,Fabaceae),Caprifoliaceae)));
TREE Fig._7 = [&R]
(Poaceae,(((Apiaceae,Chenopodiaceae),Caprifoliaceae),(Brassicaceae,Fabaceae)));
TREE Fig._8A = [&R]
(((Apiaceae,Chenopodiaceae),(Brassicaceae,Fabaceae),Caprifoliaceae),Poaceae);
TREE Fig._8B = [&R]
((Apiaceae,Brassicaceae,Caprifoliaceae,Chenopodiaceae,Fabaceae),Poaceae);
```

Fig. 3.3 String Representation of a Phylogenetic Tree

The formatted tree representation, in the form of online search result, is shown in the Fig. 3.3. For the tree named "TREE Fig._6", with reference to Fig. 3.3, is as shown below:

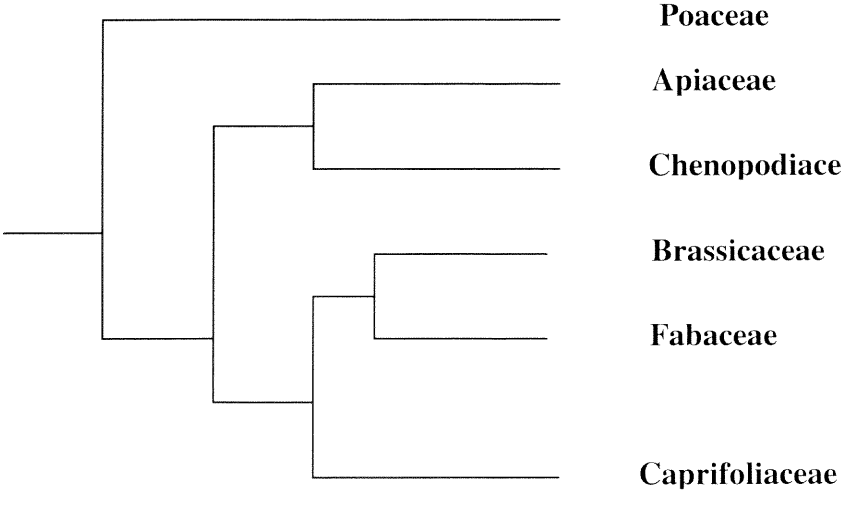

Fig. 3.4 Formatted Phylogenetic Tree Representation.

An example of the non-related data (noise), in the form of data matrices of the phylogenetic sequences is shown in the figure below.

```
BEGIN CHARACTERS;
DIMENSIONS  NCHAR=28;
FORMAT SYMBOLS= " 0 1 2" MISSING=9 GAP=- ;
MATRIX
[                    10       20      ]
[                    .        .       ]

Dalbergia        0000000000000000000000000000
Kunstleria       0000001000011100000000010100
Lonchocarpus       1000021011111110010000110110
Piscidia         1010021011111110010000110110
Tephrosia         1110021111111111011000111110
Sphinctospermum   1111112110001001111000091911
Siesbania         00i0011001000000011000000110
Hebestigma         0000110010000001911000110111
Robinia          0000110110000001111111111111
Peteria          0001110110000001111111111111
Coursetia         0000112110000001111011111111
;
END;
```

Fig. 3.5 Sample of Non-Related Data in the Input Files

Keeping in view the requirement of the local TreeSearch, there is lot of noise in the data present in Harvard ftp site. There are as many as 947 (as of today) files, each consisting of information pertaining to the study carried out and also the results including the data matrices and the phylogenetic trees.

3.2 Input to the Tool

An idea of the relevant data that is processed by the tool is given by presenting the content of a file, in Appendix C. The part of the input file that is being processed is

presented in *italicized* format. The strings containing numbers in brackets, at the end of the file, represent phylogenetic data in phylogenetic tree format. The file also contains the required phylogenetic terms, which correspond to the numbers in the string mentioned above.

3.3 Output

The processed output of the tool is the string representation of all the phylogenetic trees along with the name of the file stored locally. The output is ready for further operations to be performed.

CHAPTER 4

THE TOOL

4.1 Overview

The technologies used to program the tool are Java, and Perl programming languages. The Awk and Perl Programming manuals have been referred for writing efficient and easily understandable pattern matching 'regular expressions' explained further in the chapter [15][16]. The scope of Perl programming in the tool is identifying & matching the patterns, to extract the string representation of the phylogenetic trees from the flat text files on the ftp site. The perl script is subsequently ported into Java using custom Java classes developed by a software company named ORO Inc. at the URL www.savarese.org [3].

The company has developed reliable custom APIs for converting most of the text processing programs on Unix to Java. The instructions to download and use the software are illustrated in the Appendix D. The tool has been programmed in Java alone for reasons of cross platform compatibility. The application is designed to be a single standalone program that can be invoked by a single command. An overview of the pattern matching capabilities of perl is presented below, before detailing about the patterns implemented, methodology employed for the data collection & cleaning and the code of the tool.

4.2 Overview of Regular Expressions for Pattern Matching

A regular expression, in perl, is a syntax implemented that facilitates numerous text processing operations including pattern matching, string comparisons, replacements

and selections, to name a few. Compiled regular expression is a data structure that can be stored once and used again and again. Pre-compiled regular expressions can be used for dynamic matches that don't need to be recompiled each time they are encountered [6][7][8][9][10][11][12][13][14].

An example of regular expression is shown in the figure 4.1 below

```
$reg = qr/foo+bar?/;  # reg contains a compiled regular expression.
```

Fig. 4.1 Sample Regular Expression

Then $reg can be used as a regular expression.

```
$x = "fooooba";
$x =~ $reg;     # matches, just like /foo+bar?/
$x =~ /$reg/;   # same thing, alternate form
```

Fig. 4.2 Usage of Regular Expressions

An Example of pre-compiled regular expression is shown in the next page.

```
#!/usr/bin/perl
   # multi_grep - match any of <number> regexps
   # usage: multi_grep <number> regexp1 regexp2 ... file1 file2 ...
   $number = shift;
   $regexp[$_] = shift foreach (0..$number-1);
   @compiled = map qr/$_/, @regexp;
   while ($line = <>) {
      foreach $pattern (@compiled) {
         if ($line =~ /$pattern/) {
            print $line;
            last;  # we matched, so move onto the next line
         }
      }
   }

   % multi_grep 2 last for multi_grep


      $regexp[$_] = shift foreach (0..$number-1);
        foreach $pattern (@compiled) {

              last;
```

Fig. 4.3 Usage of Pre-Compiled Regular Expressions

Storing pre-compiled regular expressions in an array '@compiled' allows us to simply loop through the regular expressions without any recompilation, thus giving flexibility without compromising speed.

4.3 Grouping in Perl Expressions

Grouping allows parts of a regular expression to be treated as a single unit. Parts of a regular expression are grouped by enclosing them in parentheses. Grouping metacharacters () also serves another completely different purpose. It allows the extraction of the parts of a string that has matched. This is very useful to find out what part of the original string matched and also for text processing in general. For each

```
/(a|b)b/;    # matches 'ab' or 'bb'
/(ac|b)b/;   # matches 'acb' or 'bb'
/(^a|b)c/;   # matches 'ac' at start of string or 'bc' anywhere
/(a|[bc])d/; # matches 'ad', 'bd', or 'cd'
```

Fig. 4.4 Example of Grouping in Regular Expressions

grouping, the part that matched inside the expression goes into the special variables $1, $2, etc. They can be used just as ordinary variables. This is shown in the Fig. 4.4 and Fig. 4.5.

```
# extract hours, minutes, seconds
$time =~ /(\d\d):(\d\d):(\d\d)/;  # match hh:mm:ss format
$hours = $1;
 $minutes = $2;
 $seconds = $3;
```

Fig. 4.5 Usage of Groups in Regular Expressions

4.4 The Regular Expressions for Identifying the File Names

Firstly, a comprehensive sampling of all the lexicographical combinations, as shown in Fig 4.6 below, of file names is collected to make the regular expression generic enough to match all the file names. After repeated perusal, unique patterns are identified. Every sample or file name is a potential pattern to be taken into consideration. The procedure is illustrated in the figure in the next page:

Sample	Reference Tag
M1004	-1-
M1013	-1-
M101c2x3x96c12c55c08	-3-
M101c2x3x96c13c11c04	-3-
M1021	-1-
M102c2x3x96c13c14c20	-3-
M1031	-1-
M101c2x3x96c12c55c08	-3-
M101c2x3x96c13c11c04	-3-
M108c2x3x96c13c54c35	-3-
M10c11x4x95c21c26c07	-4-
M142c2x4x96c17c14c58	-5-
M14c11x5x95c11c38c19	-4-
M156c9x6x96c10c39c16	-3-
M160c11x16x96c21c09c01	-3-
M161c11x16x96c22c10c59	-3-
M164c1x6x97c14c48c13	-5-
M38c11x5x95c20c13c47	-4-
M3c11x4x95c20c31c27	-6-
M7c11x4x95c21c02c27	-6-
M80c1x29x96c17c16c17	-7-
M6c11x4x95c20c55c23	-6-
M60c1x28x96c14c00c15	-7-
M4c11x4x95c20c41c09	-6-
M1037	-1-
M999	-2-
M108c2x3x96c13c54c35	-5-
M831.NX	-8-
M833.NX	-8-
M835	-2-

Note: All the samples with same Reference tags have similar combination of numbers and characters in their corresponding relative positions.

Fig. 4.6 Sample of Lexicographical Combinations

The samples projecting unique patterns in the above sampling are shown in the figure shown in the next page:

Sample	Reference Tag
M1013	-1-
M835	-2-
M161c11x16x96c22c10c59	-3-
M10c11x4x95c21c26c07	-4-
M164c1x6x97c14c48c13	-5-
M4c11x4x95c20c41c09	-6-
M60c1x28x96c14c00c15	-7-
M833.NX	-8-

Observation: Every sample is different from another in character and number combination.
The regular expressions need to be generic enough to match all the unique patterns.

Fig. 4.7 Unique Patterns

The regular expression that match the above identified patterns are illustrated in the picture below:

1. M\d{1,3}\D\d{1,2}\D\d{1,2}\D\d{1,2}\D\d{1,2}\D\d{1,2}\D\d{1,2}

2. M\d{3,4}

3. M\d{3,4}\ .\D{2}

Note:
The regular expression '1' can match patterns with reference tags '3' through '7'.
The regular expression '2' can match patterns with reference tags '1' and '2'.
The regular expression '3' can match patterns of reference tag '8'.

The number of regular expressions depends upon the programming considerations and the text processing requirements.

Fig. 4.8 Regular Expressions for the Patterns Identified

The perl script files used for processing the files are listed in Appendix B.

4.5 Regular Expression for String Representation of Phylogenetic Data

4.5.1 Data Cleaning Overview

Powerful text processing capabilities of perl language facilitate quick and efficient data cleaning. Efficient processing of data has been accomplished by regular expressions that do string comparisons, replacements and selections. Utilizing the 'options' in regular expressions enhances the text processing capability of the expressions.

4.5.2 Data Cleaning Regular Expressions

While processing the flat files, we require to get rid of numerous white spaces, tab spaces new-line characters, return characters and form feed characters. Firstly the block of data to be processed is identified and collected into a convenient data structure for ease of retrieval and processing. The block of data, in the flat file, of interest starts from the word "TRANSLATE" and ends after the last line containing "TREE".

The block of data of interest will be referred further as *data*. *data* consists of phylogenetic terms and tree structure. The sub-block of *data*, containing the phylogenetic trees, is cleaned off all the white space characters. The data structure consists of a vector containing the phylogenetic terms in the appropriate positions. The sub-block of the *data* containing the phylogenetic trees is also clean for the white space characters and stored in a vector for convenient search and retrieval. A *for* loop is used to loop through the elements of the vector containing phylogenetic data and during the looping operation; every tree is checked for all the phylogenetic terms. This is done by trying to search for a match of the 'number', against the phylogenetic trees, in every tree and replacing the match with the corresponding phylogenetic term, to produce the

string representation of the trees. Please find the perl script for producing the string representation in the Appendix B.

4.6 Methodology for Data Collection and Cleaning

Java networking classes [2] and the classes made by ORO Inc at www.savarese.org have been extensively used for reliable data collection and cleaning. The powerful pattern matching and text processing capabilities of perl are utilized to arrive at a methodology to clean the data.

Firstly, a program is written in perl to carry out the processing of the input file. It is checked and cross-verified for its reliability with numerous files from the main ftp site. Then the perl program in converted into a standalone java program for further integration with the program that gathers the URLs of the file names. An overview of the methodology employed for gathering and cleaning of data is illustrated in the following three figures in the following pages.

The program, incorporating the custom java classes developed by ORO Inc and the native Java classes, implementing the above methodology is listed in the Appendix A. The tool has been flexibly programmed, dividing it into mainly the filename collecting part and the later recursive data collecting and cleaning module. Each of the modules can be conveniently updated according to the changing requirements. It is appropriately commented for effortless review.

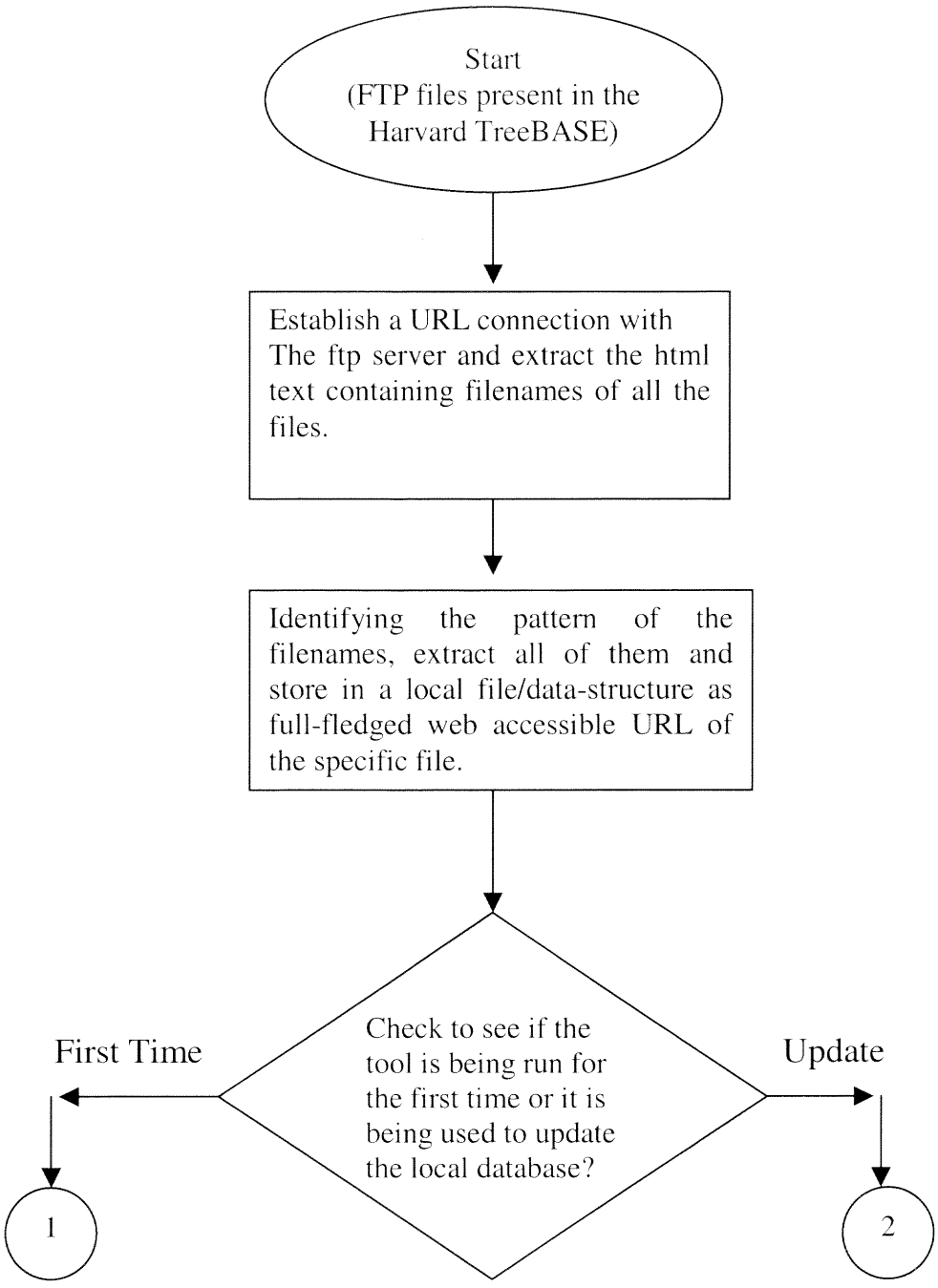

Fig. 4.9 Methodology -1

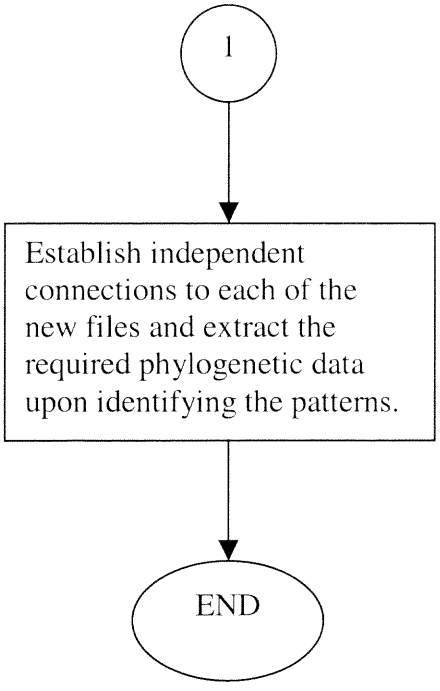

Fig 4.10 Methodology -2

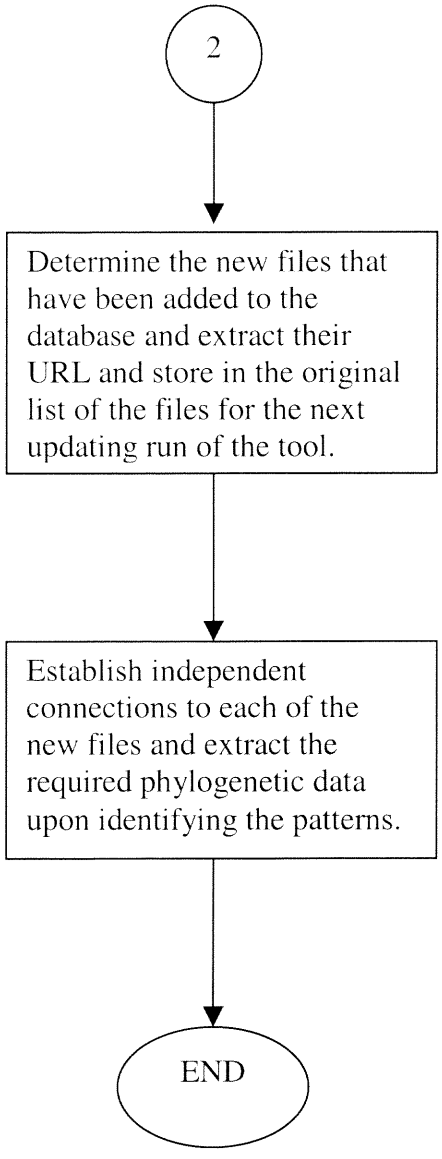

Fig. 4.11 Methodology -3

4.7 Summary

The program, incorporating the custom Java classes developed by ORO Inc and the native Java classes, implementing the above methodology is listed in the Appendix A. The tool has been flexibly programmed, dividing it into mainly the filename collecting part the later recursive data collecting and cleaning module. Each of the modules can be conveniently updated according to the changing requirements. It is appropriately commented for effortless review.

CONCLUSION

During the entire research, it was observed that numerous patterns of data exist for processing. To process a variety of patterns, the regular expressions of the processing tool used or the processing data structures for any other kind of phylogenetic data. require being generic.

The most interesting part of this thesis effort is the perl regular expressions that account for variety of patterns in the input files and make the tool a real world application. The tool can be constantly upgraded and made good for latest patterns in the input files by updating the regular expression and keeping the rest of the application intact. The standard java networking classes and also the well-tested custom java classes developed by www.savarese.org make the tool robust and platform independent.

There is enormous scope for research in the field of Phylogenetics. Building efficient databases of the life forms is a great challenge because the any such database keeps growing with our research and deeper understanding of the life forms on our planet. There is a great need to develop efficient data-structures for efficient search and retrievals from the databases and also to facilitate the advanced form of query and retrieval based on structure matching.

APPENDIX A

CODE LISTING FOR THE PHYLOGENETIC DATA CLEANING AND SEARCHING TOOL

Here are the guidelines for usage of the Phylogenetic Data Cleaning and Searching tool.

1. The tool has been programmed completely in Java.
2. User just needs to have some basic knowledge about java runtime environment.
3. User can follow the instructions illustrated on the company's website.
4. The custom classes require to be placed in appropriate directory in the system classpath and then the program is to be complied.
5. The compilation process gives an executable (". class") file. The files requires to be run at the appropriate system prompt (DOS / UNIX).

```
import java.io.*;
import java.util.*;
import com.oroinc.text.regex.*;
import com.oroinc.text.perl.*;
import com.oroinc.text.*;
import java.net.*;

public class integrate2{
//module 1 for collection of file names
        public static void main(String args[]){
        URL ftpurl = null,ftpfileurl = null;
        int maincount=0;
        Date now = null;
        Calendar hello= Calendar.getInstance();
        now = hello.getTime();
        System.out.println("Starting at "+ now.toString()+"\n");
        String reference = "ftp://herbaria.harvard.edu/pub/piel/Data/files/";
        StringBuffer urlbuffer = new StringBuffer();

//buffer for the main FTPURL
```

```
StringBuffer strbuf = new StringBuffer();
String hi = null,temp = null, trystring,data1 = null, data2 = null,
temporary = null, tstring = null;
StringSubstitution sub = new StringSubstitution("");
StringSubstitution numsub = new StringSubstitution(" ");

Vector vec = new Vector();
Vector phyldata = new Vector();
Vector treedata = new Vector();
Vector result = new Vector();
Vector strrep = new Vector();
Vector tempvec;
Vector replacement = new Vector();

try{
 ftpurl = new URL("ftp://herbaria.harvard.edu/pub/piel/Data/files");
}catch(MalformedURLException mue){
System.out.println(mue.toString()+"howdy mue\n");
}

PatternCompiler compiler = new Perl5Compiler();
Perl5Matcher match = new Perl5Matcher();
PatternMatcherInput inp,input;
MatchResult mresult=null;

Perl5Pattern pattern1 =  null,pattern = null,splitpatt = null,pattern =
null,stringpatt = null;,subpattern=null,pattern3=null;
try{
      pattern1 =
(Perl5Pattern)compiler.compile("M\\d{1,3}\\D\\d{1,2}\\D\\d{1,2}\\D\\d{1,
2}\\D\\d{1,2}\\D\\d{1,2}\\D\\d{1,2}",Perl5Compiler.CASE_INSENSITIV
E_MASK | Perl5Compiler.MULTILINE_MASK);

      pattern2 =
(Perl5Pattern)compiler.compile("M\\d{3,4}",Perl5Compiler.CASE_INSE
NSITIVE_MASK | Perl5Compiler.MULTILINE_MASK);

      pattern3 =
(Perl5Pattern)compiler.compile("M\\d{3,4}\\.\\D{2}",Perl5Compiler.CAS
E_INSENSITIVE_MASK | Perl5Compiler.MULTILINE_MASK);

      pattern =  (Perl5Pattern)compiler.compile("[
\\s\t\n]",Perl5Compiler.CASE_INSENSITIVE_MASK |
Perl5Compiler.MULTILINE_MASK);
```

```
            splitpatt =  (Perl5Pattern)compiler.compile("[\\s\t\n\f\r
]",Perl5Compiler.CASE_INSENSITIVE_MASK |

Perl5Compiler.MULTILINE_MASK);

            subpattern =
(Perl5Pattern)compiler.compile("\\d+",Perl5Compiler.CASE_INSENSITI
VE_MASK | Perl5Compiler.MULTILINE_MASK);
        }catch(MalformedPatternException MPE){
        System.out.println("howdy..malformed
        pattern"+MPE.toString()+"\n");
        }

         try{
        File urlfile = new File("hi.txt");
        File outfile = new File("mainop.txt");
        RandomAccessFile raf = new RandomAccessFile(urlfile,"rw");
        BufferedReader in = new BufferedReader(new InputStreamReader
(ftpurl.openStream()));
        hi = in.readLine();
        // to carry out the main pattern search and collecting the file names
        //and storing it to a local file.
        while(hi != null){
        inp = new PatternMatcherInput(hi);
        urlbuffer = urlbuffer.append(reference);
            if(match.contains(inp,pattern1)){
            mresult = match.getMatch();

            strbuf.append(mresult.toString());
            }else{
                inp.setCurrentOffset(inp.getBeginOffset());
                if(match.contains(inp,pattern3)){
                    mresult = match.getMatch();
                    strbuf.append(mresult.toString());
                }else{
                inp.setCurrentOffset(inp.getBeginOffset());
                    if(match.contains(inp,pattern2)){
                    mresult=match.getMatch();
                    strbuf.append(mresult.toString());
                    }//end of if
                }//end of else
            }
            urlbuffer = urlbuffer.append(strbuf.toString());
            if(!(urlbuffer.toString().equals((Object)reference))){
            maincount++;
            System.out.println("file name"+ maincount +"being stored : - )
```

```
            \n");

            raf.writeBytes(urlbuffer.toString()+"\n");
            }

            hi = in.readLine();
            strbuf=strbuf.delete(0,strbuf.length());
            urlbuffer=urlbuffer.delete(0,urlbuffer.length());

      }//end of outer while

      //file name extraction  ends here...END of MODULE 1

      System.out.println("I am done with file name extraction !!\n");

       //leaving a time stamp

      hello= Calendar.getInstance();
      now = hello.getTime();
      System.out.println("File Names Extracted at"+ now.toString()+"\n");
      RandomAccessFile ram = new RandomAccessFile(urlfile,"r");
      RandomAccessFile ramf = new RandomAccessFile(outfile,"rw");
      maincount=0;

      //MODULE 2

      temp = ram.readLine();
      while(temp != null){

      //while loop for extraction and cleaning of files recursively
      //the main data collection and cleaning starts here--files being
      //individually extracted
            // ------------the integration begins here---------**********
            maincount++;
                  System.out.println("file number"+ maincount +"being
processed : - ) \n");
                  //raf.writeBytes(urlbuffer.toString()+"\n");
                  ramf.writeBytes(temp+"\n");
                  //System.out.println(urlbuffer.toString()+"\n");
                  try{
                  //ftpfileurl = new URL(urlbuffer.toString());
                        ftpfileurl = new URL(temp);
                  }catch(MalformedURLException mue){
                  System.out.println(mue.toString()+"howdy ftpfile \n");
                  }
```

```
        BufferedReader ftpfilein = new BufferedReader(new
InputStreamReader (ftpfileurl.openStream()));

                String innerstring = ftpfilein.readLine();
                 //data being read from the particular ftp file
                int linecount=0;

                while(innerstring != null){
                  linecount++;
                  trystring = Util.substitute(match,pattern,sub,innerstring);
                  vec.add((Object)trystring);
                  //System.out.println("vec elementis "+trystring+"\n");
                  innerstring = ftpfilein.readLine();
                }

                ftpfilein.close();
                 //the connection with the particular file is closed
                //System.out.println("the size of the vec is
"+vec.size()+"\n");
                System.out.println("i just closed the file connection\n");
                //setting flags

                int startmark=0,flag=0,lastmark=0,limit =
Util.SPLIT_ALL,sublimit=Util.SUBSTITUTE_ALL;

                  for(int i=0;i<linecount;i++){
if(vec.elementAt(i).toString().indexOf("TRANSLATE") != -1){
                      startmark = i;
                     }
                  }//end for

                  for(int i=startmark;i<linecount;i++){
                    if((vec.elementAt(i).toString().indexOf(";") != -1)&&(flag
== 0)){
                      lastmark = i;
                      flag = -1;
                    }
                    //treemark = vec.elementAt(i).toString().indexOf("TREE");
                    if(vec.elementAt(i).toString().indexOf("TREE") != -1){
                       treedata.add((Object)vec.elementAt(i).toString());
                     }
                   }

                  for(int i=startmark+1;i<lastmark;i++){//the main loop
                      phyldata.add(vec.elementAt(i));
```

```
        result = Util.split(match,splitpatt,vec.elementAt(i).toString(),limit);

                            for(int c=0;c<result.size();c++){
                             if(c==1){
                              data1 = result.elementAt(c).toString();
                             }
                             if(c==2){
                              data2 = result.elementAt(c).toString();
                              StringBuffer tempbuf = new
StringBuffer(data2);
                              int index = data2.indexOf(",");
                              if(index != -1){
                                tempbuf = tempbuf.deleteCharAt(index);
                              }
                              numsub.setSubstitution(tempbuf.toString());
                              tempbuf.delete(0,tempbuf.length());
                            }
                            }//end for

                        StringBuffer strbuffer = new StringBuffer("([,\)\(])");
                        strbuffer.append(data1);
                        strbuffer.append("([,\)\(])");

                        try{
                             stringpatt =
(Perl5Pattern)compiler.compile(strbuffer.toString(),Perl5Compiler.CASE_
INSENSITIVE_MASK | Perl5Compiler.MULTILINE_MASK);
                        }catch(MalformedPatternException mpe){
System.out.println("MPE Exception"+mpe.toString()+"\n");
                             System.exit(1);
                        }
                        tempvec = new Vector();

                        for(int b=0;b<treedata.size();b++){ StringBuffer
tempbuffer = new StringBuffer(treedata.elementAt(b).toString());
                             if(tempbuffer.toString().indexOf("]") != -1){
                              tstring =
tempbuffer.toString().substring(0,tempbuffer.toString().indexOf("]")+1);
                              tempbuffer =
 tempbuffer.delete(0,tempbuffer.toString().indexOf("]")+1);
                              replacement.add((Object)tstring);
                             }
                           input = new
PatternMatcherInput(tempbuffer.toString());
```

```
        while(match.contains(input,stringpatt)){//the while
                                mresult = match.getMatch();
                                //ram.writeBytes("the item is
\t"+mresult.toString()+"\n");

Util.substitute(match,subpattern,numsub,mresult.toString());
//ram.writeBytes("the item is \t"+temporary.toString()+"\n");

                tstring = input.toString();
                StringSubstitution newsub = new
StringSubstitution(temporary);
                                tstring =
Util.substitute(match,stringpatt,newsub,tstring);
                                //ram.writeBytes("the item is
\t"+temporary.toString()+"\n");
                                tempvec.add((Object)tstring);
                                //System.out.println(tstring+"\n");
                                //ram.writeBytes("the replacement's size
is"+replacement.size()+"\n");

                                };//while ends here

                            }//treedata for loop ends here

                            if(tempvec.size()==treedata.size()){
                              for(int j=0;j<tempvec.size();j++){
                                 treedata.setElementAt(tempvec.elementAt(j),j);
                              }
                            }
                            tempvec.clear();
                           }//end of main for looooooop

                           vec.clear();

                              //System.out.println("i am past the main forlooop\n");
                            if(replacement.size()==treedata.size()){
                                for(int z=0;z<treedata.size();z++){
                                StringBuffer buffer1 = new
StringBuffer(replacement.elementAt(z).toString());
buffer1.append(treedata.elementAt(z).toString());
treedata.setElementAt((Object)buffer1.toString(),z);
                                }
                            }
                            for(int i=0;i<treedata.size();i++){
ramf.writeBytes(treedata.elementAt(i).toString()+"\n");
```

```
                    }
                ramf.writeBytes("file break ----------\n");
                hello= Calendar.getInstance();
                now = hello.getTime();
                System.out.println("now it is "+ now.toString()+"\n");
                treedata.clear();
                result.clear();
                strrep.clear();
                replacement.clear();
        //integration ends here--------
        temp = ram.readLine();
        }
        }catch(IOException ioe){
        System.out.println(ioe.toString()+"howdy io\n");
        }
        //System.out.println("the whole match is !\n"+strbuf.toString()+"\n");
        }
}
```

APPENDIX B

CODE LISTINGS

Two code listings containing the perl shell scripts are presented for reference. These text-processing scripts are later converted into java.

Code Listing 1

This perl script takes the flat file from the ftp-site as input and produces a file containing the string representation and simultaneously, displays the intermediate processing data-structures to the user.

```
#!opt/local/bin/perl
unless(open(howdy,"$ARGV[0]")){
die("howdy..no luck");
}

@mainarray=<howdy>;
$lines=@mainarray;

#       print("number of lines in t $lines");

//identifying the start of data block

unless(open(howdy2,"grep -n \"TRANSLATE\" $ARGV[0] |")){
die("failed at second howdy");
}

//cleaning the input for any non-digit characters to extract the start line number

$string = <howdy2>;
$string =~ s/[\D ];
$string = $string +0;

#       print("\n \$string  is $string \n");
close(howdy2);

#---------

//cleaning the input for unnecessary space characters
```

```
for($minicount=$string;$minicount <= $lines;$minicount++){
        $variab=$mainarray[$minicount];
        # prnt("$variab");
        $variab =~ tr/^[ \t]//d;
        $variab =~ tr/[ \t\n]$//d;

#to check  for ";" pattern

        if($variab =~ /[;]/){

                $note = $minicount;
#       print("\n required number is $note \n");
        last;
        }

}


#       print("value of note is $note");

#---------

$endlimit = $note;

#       print("\n the new limits are $string and $endlimit \n");

#       $count=$string;
        $datacount=0;

        for($count=$string;$count <= $endlimit; $count++){
                @dataarray[$datacount++]=$mainarray[$count];
        #       eval("print (\"$mainarray[$count]\")>newfile ;");
        }
        print("The phylogenetic data is \n @dataarray \n");

#       print("chal see you.things done\n ");

$arraycount=0;
#this loop will colect the trees into the treearray,assuming that the TREE and tree
are in
//a single line.
foreach $variable (@mainarray){
If ($variable =~ /\bTREE\b/ ){

#       print ("\n \$variable is $variable\n");
@treearray[$arraycount++]=$variable;
```

```
}
}
print("\nTree array is \n @treearray \n");

#-----------------------------**********************
#         @try = <file>;


#         print "the contents of the first file are\n @try \n";

@dataarray2 = @dataarray;
@treearray2 = @treearray;

$temp1 = @dataarray2;
#       print "the temp1 is $temp1\n";

#         @try1 = <file2>;
#         print "contents of the second file are \n @try1 \n";
$temp2 = @treearray2;
#       print "the temp2 is $temp2 \n";


$index1 = 0;
while($index1 < $temp1){
#       print "contents are $dataarray2[$index1]\n";
#       $index1++;
        $temp2index =0;
        chop($dataarray2[$index1]);
        @temp = split(/[\t\n\f\r ]+/,$dataarray2[$index1]);
        if($temp[0] eq "")
        {
        shift(@temp);
        }
        $tsize = @temp;
        $tempindex = 0;
#       while($tempindex < $tsize){
#       print "the word is $temp[$tempindex] \n";
#       $tempindex++;
#       }
        chop($temp[1]);
        while($temp2index < $temp2){
        $treearray2[$temp2index++] =~ s/([,\)\(])$temp[0]([,\)\(])/$1$temp[1]$2/;
        #$temp2index++;
        }

#       $var = $index *2;
```

```
#       @array[$var,$var+1]=$temp[0,1];
#       print "temp 1 is $temp[1]\n";
        $index1++;
}
#print "the changed arrays are @try1 \n";
$tempindex=0;


unless(open(new,">newfile2")){
die("cannot open a new file bye\n");
}

while($tempindex < $temp2)
{
print new "\n $treearray2[$tempindex] \n";
$tempindex++;
}
```

Code Listing 2

This perl script has been used to test the regular expression for matching the file names of the flat files.

```
#!opt/local/bin/perl
#print("hi  the match is ");
$input = "src=\"doc:/lib/images/ftp/file.gif\" border=0 width=24
height=26>      M101c2x3x96c12c55c08";
$count=1;
while($input =~
/M\d{1,3}\D\d{1,2}\D\d{1,2}\D\d{1,2}\D\d{1,2}\D\d{1,2}\D\d{1,2}/){
$match = $match + $&;
print ("the match is $match \n");
$count +=1;
}
```

APPENDIX C

SAMPLE INPUT FLAT-FILE

The data that is processed by the program is italicized in the sample flat-file listed below. The italicized block of the file consists of phylogenetic terms and also the representation of the phylogenetic tree.

#NEXUS

[MacClade 3.05 registered to MICHAEL DONOGHUE,
HARVARD]
BEGIN DATA;
DIMENSIONS NTAX=28 NCHAR=14;
[!This data set was downloaded from TreeBASE, a prototype relational database of phylogenetic knowledge. TreeBASE has been supported by the NSF, Harvard University, and UC Davis. Please do not remove this acknowledgment from the Nexus file.
TreeBASE © 1994-1996.
Study reference:
Soltis, D. E., P. S. Soltis, and B. D. Ness. 1989. Chloroplast-DNA variation and multiple origins of autopolyploidy in Heather micranthia (Saxifragaceae). Evolution 43:650-656.
Study accession number = S2x3x96c17c06c51
Matrix accession number = M115c2x3x96c17c09c29
]
FORMAT MISSING=? GAP=- ;
MATRIX
[10]
[.]
Heuchera_micrantha_P_404 00000000000000
Heuchera_micrantha_P_1555 00000000000000
Heuchera_micrantha_P_1694 00000000000000
Heuchera_micrantha_P_1693 00000000100000
Heuchera_micrantha_P_357 00000000000010
Heuchera_micrantha_P_354 00000000000000
Heuchera_micrantha_D_344 00000000000001
Heuchera_micrantha_D_343 00000000000000
Heuchera_micrantha_D_1736 00000000000001

```
Heuchera_micrantha_D_325    11110000000000
Heuchera_micrantha_D_324    00000000000001
Heuchera_micrantha_D_400    00000000000000
Heuchera_micrantha_D_1572   00000000000000
Heuchera_micrantha_D_398    11110000000000
Heuchera_micrantha_D_397    11110000000000
Heuchera_micrantha_D_1814   00000000000000
Heuchera_micrantha_D_1809   00000000000001
Heuchera_micrantha_D_1767   00000000000001
Heuchera_micrantha_E_407    00000110100000
Heuchera_micrantha_E_1542   00000110100000
Heuchera_micrantha_E_389    00000010100000
Heuchera_micrantha_E_1713   00000000000001
Heuchera_micrantha_H_1558   00001001000000
Heuchera_micrantha_H_1949   00000000000000
Heuchera_micrantha_M_1578   00000000001100
Heuchera_micrantha_M_1641   00000000010000
Heuchera_micrantha_M_1570   00000000001100
Heuchera_micrantha_M_1567   00000000000000
;
END;
BEGIN ASSUMPTIONS;
        OPTIONS  DEFTYPE=unord PolyTcount=MINSTEPS ;
END;
BEGIN TREEBASE;
END;
BEGIN TREES;
[!1 trees, starting with  TreeBASE accession#: T295:2/3/96:17:10:24]
        TRANSLATE
                1       Heuchera_micrantha_P_404,
                2       Heuchera_micrantha_P_1555,
                3       Heuchera_micrantha_P_1694,
                4       Heuchera_micrantha_P_1693,
                5       Heuchera_micrantha_P_357,
                6       Heuchera_micrantha_P_354,
                7       Heuchera_micrantha_D_344,
                8       Heuchera_micrantha_D_343,
                9       Heuchera_micrantha_D_1736,
                10      Heuchera_micrantha_D_325,
                11      Heuchera_micrantha_D_324,
                12      Heuchera_micrantha_D_400,
                13      Heuchera_micrantha_D_1572,
                14      Heuchera_micrantha_D_398,
                15      Heuchera_micrantha_D_397,
```

```
                16      Heuchera_micrantha_D_1814,
                17      Heuchera_micrantha_D_1809,
                18      Heuchera_micrantha_D_1767,
                19      Heuchera_micrantha_E_407,
                20      Heuchera_micrantha_E_1542,
                21      Heuchera_micrantha_E_389,
                22      Heuchera_micrantha_E_1713,
                23      Heuchera_micrantha_H_1558,
                24      Heuchera_micrantha_H_1949,
                25      Heuchera_micrantha_M_1578,
                26      Heuchera_micrantha_M_1641,
                27      Heuchera_micrantha_M_1570,
                28      Heuchera_micrantha_M_1567
        ;
        TREE Fig._4 = [&R]
((1,12,6,8,16,22,13,3,2,24,28),(11,7,18,9,17),(27,25),26,5,23,(14,15,10),(4,((19,20),21)));
END;
BEGIN MACCLADE;
Version 3.05;
LastModified -1370107628;
Singles 1000&/0;
END;
-------------------------------------------------------------------------------
```

APPENDIX D

OVERVIEW OF ORO SOFTWARE

ORO, Inc. was a software development company that focused on providing high quality object-oriented class libraries and toolkits for software developers. It operated for 1.5 years. Currently the ORO name lives on as a www.savarese.org project. ORO (Original Reusable Objects) is distributed as open source with liberal binary licensing. The software mainly consists of text processing tools like OROMatcher 2.0, PerlTools 2.0, Awk Tools and Text Tools that are available for free download.

The guidelines for installation of the software are illustrated on the specified internet resource. The latest versions, of most of the software, support java 1.1 later upgrades of java.

REFERENCES

[1] Jiawei Han and Micheline Kamber, Data Mining,
Morgan Kaufmann Publishers, August 2000.

[2] Java(TM) 2 Platform, Standard Edition, v1.2.2 API
http://java.sun.com/products/jdk/1.2/docs/api/index.html (June - present).

[3] Daniel F. Savarese, www.savarese.org (June - September 2001).

[4] Google 2001, http://www.google.com/search?hl=en&q=%22phylogenetics%22
(02 June 2001).

[5] Michael Donoghue, William Piel, Mike Sanderson and Mary Walsh,
TreeBASE, A Database of Phylogenetic Knowledge
http://www.herbaria.harvard.edu/treebase/ (January - present).

[6] *PERLRE, Perl regular expressions*,
http://www.mit.edu:8001/perl/perlre.html, (June - August 2001).

[7] *PERL-Practical Extraction and Reporting Language*
http://www-2.cs.cmu.edu/People/rgs/perl.html, (June - August 2001).

[8] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger
The AWK Programming Language Addison-Wesley, 1988.

[9] David Till, May 1997. Teach Yourself Perl 5 in 21 days, O'reilly Publications
(May - September 2001).

[10] Carlos Ramirez, (17 September 2001) Perl 5.6.1 Documentation,
http://www.perldoc.com/perl5.6.1/pod/perlretut.html, (June - August 2001).

[11] Fermi National Accelarator Laboratory,
perlref - Perl references and nested data structures
http://www.fnal.gov/docs/products/perl/pod.new/5.00503/pod/perlref.html
(June 2001).

[12 Nick Moraitakis & Giorgos Zervas, (1999). Perl Reference
http://www.perlreference.com/lang/ (August 2001).

[13] O'Reilly & Associates, Inc. Perl.com, The source of perl (1998-2001)
http://language.perl.com/faq/, (August 2001).

[14] Kent Landfield, Internet FAQ Archives, Perl FAQ
http://www.faqs.org/faqs/perl-faq/, (August - September 2001).

[15] Arnold D. Robbins, et al, AWK Programming Language http://www.cl.cam.ac.uk/texinfodoc/gawk_toc.html#SEC2 (June 2001).

[16] Nicholas Reynolds, et al, Help-Site Computer Manuals http://help-site.com/c.m/prog/lang/awk/ (June 2001), AWK Programming.

[17] Rich Grenyer, et al, The Purvis Lab, Imperial College http://www.bio.ic.ac.uk/evolve/, (September 2001).

[18] David R. Maddison and W. P. Maddison, (2001) MacClade 4 Home Page, http://phylogeny.arizona.edu/macclade/macclade.html, (September 2001).

[19] Joe Felsenstein, PHYLIP, http://evolution.genetics.washington.edu/phylip.html (September 2001).

[20] Huiyuan Shan and Dr. Jason T.L. Wang, TreeSearch http://aria.njit.edu/~biotool/search_index.html, (September 2001).

[21] Roderic D. M. Page, et al, (4 July 2000) http://taxonomy.zoology.gla.ac.uk/rod/treemap.html (September 2001).

[22] Maidak BL, Cole JR, Lilburn TG, Parker CT Jr, Saxman PR, Farris RJ, Garrity GM, Olsen GJ, Schmidt TM, Tiedje JM. The RDP-II (Ribosomal Database Project). *Nucleic Acids Res* 2001 Jan 1;29 (1):173-4 http://rdp.cme.msu.edu/html/index.html, (September 2001).

[23] Antje Krause, et al, (17 September 2001), SYSTERS Cluster Set Protein Database Release 3 http://systers.molgen.mpg.de/ (September 2001).

[24] David R. Maddison and Wayne P. Maddison, (1998) The Tree of Life http://phylogeny.arizona.edu/tree/phylogeny.html (September 2001).